DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2018

Scalable System-Wide Traffic Flow Predictions Using Graph Partitioning and Recurrent Neural Networks

JÓN REGINBALD ÍVARSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Scalable System-Wide Traffic Flow Predictions Using Graph Partitioning and Recurrent Neural Networks

JÓN REGINBALD ÍVARSSON

Master's program in Software Engineering of Distributed Systems
Date: December 20, 2018
Supervisor: Zainab Abbas and Ahmad Al-Shishtawy
Examiner: Vladimir Vlassov
Swedish title: Skalbara systemövergripande trafikprognoser med grafpartitionering och återkommande neurala nätverk
The School of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I would like to dedicated this thesis to my parents, Ívar Jónsson and Lilja Mósesdóttir, whose encouragement and support was vital while pursuing my academic ambitions. I would like to express my sincere gratitude to my supervisors, Zainab Abbas and Ahmad Al-Shishtawy, and my examiner, Vladimir Vlassov. Their guidance, ideas and knowledge proved invaluable throughout the thesis work. Special thanks to RISE SICS, Research Institutes of Sweden, and KTH, the Royal Institute of Technology in Stockholm Sweden, for giving me the opportunity to work on this project. Finally, I would also like to thank my friends and classmates for their support and my opponents, Þorsteinn Þorri Sigurðsson and Cosar Ghandeharioon, for their helpful feedback during the review of the thesis.

Reykjavík Iceland, December, 2018
Jón Reginbald Ívarsson

Abstract

Traffic flow predictions are an important part of an Intelligent Transportation System as the ability to forecast accurately the traffic conditions in a transportation system allows for proactive rather than reactive traffic control. Providing accurate real-time traffic predictions is a challenging problem because of the nonlinear and stochastic features of traffic flow. An increasingly widespread deployment of traffic sensors in a growing transportation system produces greater volume of traffic flow data. This results in problems concerning fast, reliable and scalable traffic predictions.

The thesis explores the feasibility of increasing the scalability of real-time traffic predictions by partitioning the transportation system into smaller subsections. This is done by using data collected by Trafikverket from traffic sensors in Stockholm and Gothenburg to construct a traffic sensor graph of the transportation system. In addition, three graph partitioning algorithms are designed to divide the traffic sensor graph according to vehicle travel time. Finally, the produced transportation system partitions are used to train multi-layered long short-term memory recurrent neural networks for traffic density predictions. Four different types of models are produced and evaluated based on root mean squared error, training time and prediction time, i.e. transportation system model, partitioned transportation models, single sensor models, and overlapping partition models.

Results of the thesis show that partitioning a transportation system is a viable solution to produce traffic prediction models as the average prediction accuracy for each traffic sensor across the different types of prediction models are comparable. This solution tackles scalability issues that are caused by increased deployment of traffic sensors to the transportation system. This is done by reducing the number of traffic sensors each prediction model is responsible for which results in less complex models with less amount of input data. A more decentralized and effective solution can be achieved since the models can be distributed to the edge of the transportation system, i.e. near the physical location of the traffic sensors, reducing prediction and response time of the models.

Keywords: Traffic Flow Prediction; Machine Learning; Recurrent Neural Network; Graph Partitioning; Big Data

Sammanfattning

Prognoser för trafikflödet är en viktig del av ett intelligent transportsystem, eftersom möjligheten att prognostisera exakt trafiken i ett transportsystem möjliggör proaktiv snarare än reaktiv trafikstyrning. Att tillhandahålla noggrann trafikprognosen i realtid är ett utmanande problem på grund av de olinjära och stokastiska egenskaperna hos trafikflödet. En alltmer utbredd använding av trafiksensorer i ett växande transportsystem ger större volym av trafikflödesdata. Detta leder till problem med snabba, pålitliga och skalbara trafikprognoser.

Avhandlingen undersöker möjligheten att öka skalbarheten hos realtidsprognoser genom att dela transportsystemet i mindre underavsnitt. Detta görs genom att använda data som samlats in av Trafikverket från trafiksensorer i Stockholm och Göteborg för att konstruera en trafiksensor graf för transportsystemet. Dessutom är tre grafpartitioneringsalgoritmer utformade för att dela upp trafiksensor grafen enligt fordonets körtid. Slutligen används de producerade transportsystempartitionerna för att träna multi-layered long short memory neurala nät för förspänning av trafiktäthet. Fyra olika typer av modeller producerades och utvärderades baserat på rotvärdes kvadratfel, träningstid och prediktionstid, d.v.s. transportsystemmodell, partitionerade transportmodeller, enkla sensormodeller och överlappande partitionsmodeller.

Resultat av avhandlingen visar att partitionering av ett transportsystem är en genomförbar lösning för att producera trafikprognosmodeller, eftersom den genomsnittliga prognoser noggrannheten för varje trafiksensor över de olika typerna av prediktionsmodeller är jämförbar. Denna lösning tar itu med skalbarhetsproblem som orsakas av ökad användning av trafiksensorer till transportsystemet. Detta görs genom att minska antal trafiksensorer varje trafikprognosmodell är ansvarig för. Det resulterar i mindre komplexa modeller med mindre mängd inmatningsdata. En mer decentraliserad och effektiv lösning kan uppnås eftersom modellerna kan distribueras till transportsystemets kant, d.v.s. nära trafiksensorns fysiska läge, vilket minskar prognos- och responstid för modellerna.

Nyckelord: Trafikprognoser; Maskininlärning; Återkommande Neuralt Nätverk; Graf Partitionering; Big Data

Contents

1 Introduction **1**
- 1.1 Problem Definition . . . 2
- 1.2 Purpose, Goals and Research Questions . . . 3
- 1.3 Methodology . . . 5
- 1.4 Delimitation . . . 6
- 1.5 Contributions . . . 7
- 1.6 Ethics . . . 7
- 1.7 Sustainability . . . 8
- 1.8 Outline . . . 9

2 Background **11**
- 2.1 Traffic Sensors . . . 11
  - 2.1.1 Inductive Loop Detectors . . . 12
  - 2.1.2 Microwave Radar . . . 13
  - 2.1.3 Video Image Processor . . . 14
- 2.2 Traffic Flow Theory . . . 15
  - 2.2.1 Performance Measures . . . 15
- 2.3 Graph Theory . . . 17
  - 2.3.1 Graph Concepts . . . 17
  - 2.3.2 Graph Separation . . . 19
- 2.4 Machine Learning . . . 20
- 2.5 Traffic Predictions . . . 24
  - 2.5.1 Naïve Methods . . . 25
  - 2.5.2 Parametric Methods . . . 27
  - 2.5.3 Non-Parametric Methods . . . 27

3 Related Work **29**

4 Implementation **33**
4.1 Datasets . . . 33
4.2 Graph . . . 37
4.3 Sequential Weight Based Graph Partitioning . . . 40
4.3.1 Forward Sequential Weight Based Graph Partitioning . . . 42
4.3.2 Backward Sequential Weight Based Graph Partitioning . . . 44
4.3.3 Overlapping Sequential Weight Based Graph Partitioning . . . 46
4.4 Neural Networks for Traffic Density Predictions . . . 49
4.4.1 Transportation System Neural Network . . . 51
4.4.2 Partitioned Transportation System Neural Network . . . 52
4.4.3 Single Sensor Neural Network . . . 53
4.4.4 Overlapping Partition Neural Network . . . 54

5 Experiments **55**
5.1 Graph . . . 55
5.2 Sequential Weight Based Graph Partitioning . . . 57
5.3 Traffic Density Predictions . . . 60
5.3.1 Hyperparameter Tuning . . . 62

6 Results **65**
6.1 Graph . . . 65
6.2 Sequential Weight Based Graph Partitioning . . . 70
6.3 Traffic Density Predictions . . . 76
6.3.1 Hyperparameter Tuning . . . 80
6.3.2 Measured and Predicted Data Evaluation . . . 87
6.3.3 Time Evaluation . . . 92
6.3.4 Root Mean Squared Error Evaluation . . . 94
6.3.5 Sensor Group Evaluation within a Partition . . . 101

7 Conclusion and Future Work **119**
7.1 Limitations . . . 121
7.2 Future Work . . . 122
7.2.1 Sensor Graph Refinement . . . 122
7.2.2 Graph Partitioning Improvements . . . 123
7.2.3 Deep Learning Exploration . . . 123
7.2.4 Model Training . . . 123

7.2.5 Parallel Training 124
7.2.6 Distributed Traffic Prediction System 124
7.2.7 Federated Learning 124

Bibliography **125**

A Source Code and Software **129**

List of Figures

1.1 Sensor statistics from Stockholm and Gothenburg 3
1.2 Hierarchy of data processing 4

2.1 Key components of an inductive loop detector 12
2.2 The two types of waveforms used in traffic detection . . 13
2.3 Relationship between flow, demand and capacity during non-congested and congested conditions 16
2.4 Graph Examples 18
2.5 Graph divided into two groups of equal sizes 20
2.6 Simple neural network 22
2.7 Recurrent neural network neuron 23
2.8 Long short-term memory neuron 24
2.9 Taxonomy of traffic prediction methods 25

4.1 Density data from four sensors in November 2016 36
4.2 Traffic sensors' sites in Sweden 36
4.3 Sensor sites in Stockholm center 37
4.4 A traffic sensor's spatial dependencies 41
4.5 Difference between forward and backward partitioning . 44
4.6 Simple example of an overlapping partition 46
4.7 Transportation System Neural Network 52
4.8 Partitioned Transportation System Neural Network . . . 53
4.9 Single Sensor Neural Network 54
4.10 Overlapping Partition Neural Network 54

5.1 An example of a straight road section in Stockholm . . . 56
5.2 An example of a complicated road interchange in Stockholm 57
5.3 Simple evaluation scenario 58
5.4 Complex evaluation scenario 59

5.5 Overlapping evaluation scenario 60

6.1 The directed graph of Trafikverket's traffic sensors' sites 66
6.2 Trafikverket's traffic sensors' site graph of Stockholm overlaid on top of a map 67
6.3 Trafikverket's traffic sensors' site graph of Gothenburg overlaid on top of a map 68
6.4 Trafikverket's traffic sensor site graph overlaid of Stockholm center on top of a map 69
6.5 Results for Stockholm when partitioning the transportation system with a weight criteria of 3 minutes 71
6.6 Results for Stockholm when partitioning the transportation system with a weight criteria of 5 minutes 72
6.7 Results for Stockholm when partitioning the transportation system with a weight criteria of 10 minutes 73
6.8 Results for Stockholm when partitioning the transportation system with a weight criteria of 20 and 30 minutes . 74
6.9 Example of a overlapping partition in Stockholm 76
6.10 Relationship between training time and average number of traffic sensors 79
6.11 Relationship between prediction time and average number of traffic sensors 80
6.12 Validation loss for transportation system model 81
6.13 Validation loss for a 20 Minute partition model 81
6.14 Validation loss for a 10 Minute partition model 82
6.15 Validation loss for a 5 Minute partition model 82
6.16 Validation loss for a 3 Minute partition model 83
6.17 Validation loss for a single sensor model 83
6.18 Relationship between model complexity and training time for the transportation system model 84
6.19 Relationship between model complexity and training time for the 20 minute partitioned transportation system model 85
6.20 Relationship between model complexity and training time for the 10 minute partitioned transportation system model 85
6.21 Relationship between model complexity and training time for the 5 minute partitioned transportation system model 86
6.22 Relationship between model complexity and training time for the 3 minute partitioned transportation system model 86

6.23 Relationship between model complexity and training time for the single sensor model 87
6.24 Comparison of measured and predicted density for the Transportation System Model 88
6.25 Comparison of measured and predicted density for the 20 Minute Partitioned Transportation System Models . . 89
6.26 Comparison of measured and predicted density for the 10 Minute Partitioned Transportation System Models . . 89
6.27 Comparison of measured and predicted density for the 5 Minute Partitioned Transportation System Models . . . 90
6.28 Comparison of measured and predicted density for the 3 Minute Partitioned Transportation System Models . . . 90
6.29 Comparison of measured and predicted density for Single Sensor Models 91
6.30 Comparison of measured and predicted density for Overlapping Partition Models 91
6.31 Average training time for each model 92
6.32 Total sequential training time for each model 93
6.33 Average prediction time for each model 94
6.34 Comparison of average RMSE between all models for all traffic sensors 96
6.35 Comparison of average RMSE between all models for all traffic sensors 97
6.36 Average RMSE of all traffic sensors for the Transportation System Model 98
6.37 Average RMSE of all traffic sensors for the 20 Minute Partition System Models 98
6.38 Average RMSE of all traffic sensors for the 10 Minute Partition System Models 99
6.39 Average RMSE of all traffic sensors for the 5 Minute Partition System Models 99
6.40 Average RMSE of all traffic sensors for the 3 Minute Partition System Models 100
6.41 Average RMSE of all traffic sensors for the Single Sensor Models 100
6.42 Average RMSE of all traffic sensors for the Overlapping Partition Models 101
6.43 Average RMSE comparison between sensor groups in the 20 Minute Partition System Models 103

6.44 Average RMSE comparison between sensor groups in the 10 Minute Partition System Models 104
6.45 Average RMSE comparison between sensor groups in the 5 Minute Partition System Models 105
6.46 Average RMSE comparison between sensor groups in the 3 Minute Partition System Models 106
6.47 Comparison of average RMSE of traffic sensors in the start group in the 20 Minute Partition System Models . . 107
6.48 Comparison of average RMSE of traffic sensors in the center group in the 20 Minute Partition System Models . 108
6.49 Comparison of average RMSE of traffic sensors in the end group in the 20 Minute Partition System Models . . . 109
6.50 Comparison of average RMSE of traffic sensors in the start group in the 10 Minute Partition System Models . . 110
6.51 Comparison of average RMSE of traffic sensors in the center group in the 10 Minute Partition System Models . 111
6.52 Comparison of average RMSE of traffic sensors in the end group in the 10 Minute Partition System Models . . . 112
6.53 Comparison of average RMSE of traffic sensors in the start group in the 5 Minute Partition System Models . . . 113
6.54 Comparison of average RMSE of traffic sensors in the center group in the 5 Minute Partition System Models . . 114
6.55 Comparison of average RMSE of traffic sensors in the end group in the 5 Minute Partition System Models . . . 115
6.56 Comparison of average RMSE of traffic sensors in the start group in the 3 Minute Partition System Models . . . 116
6.57 Comparison of average RMSE of traffic sensors in the center group in the 3 Minute Partition System Models . . 117
6.58 Comparison of average RMSE of traffic sensors in the end group in the 3 Minute Partition System Models . . . 118

List of Tables

4.1 The schema of the metadata dataset 34
4.2 The schema of the traffic sensor measurement dataset . . 35
4.3 Number of LSTM hidden units 53

5.1 Number of LSTM units selected for hyperparameter tuning 63

6.1 The schema of the vertex dataset 65
6.2 The schema of the edge dataset 66
6.3 Graph statistics 70
6.4 Partitioning statistics 75
6.5 Overlapping partitioning statistics 76
6.6 Overview of the experiments conducted 77
6.7 Overview of the experimental results 78
6.8 The number of LSTM hidden units producing the most accurate traffic predictions 84
6.9 Root mean squared error difference between models for traffic sensor E265O-4215-1 87
6.10 Average RMSE of predictions for all traffic sensors 95
6.11 Median of the average traffic prediction RMSE for each traffic sensor group 102

Chapter 1

Introduction

In the past century, increased urbanization and motorization has resulted in an improved standard of living for the world population. However, significant traffic-related challenges have occurred such as traffic accidents, air quality deterioration, increased transportation cost and traffic congestion [1].

The United States Bureau of Transportation Statistics estimated in its annual report from 2017 [2] that the nation's transportation assets is valued to be approximately $7.7 trillion in 2016 and the road system contains 6,7 million kilometers of road. Since the year 2000, the roads increase by 351 thousand kilometers. The bureau found that people using cars traveled around 6.1 trillion kilometers in 2015, an increase of 5.4 percent from 2010. Registered vehicles increased by 38 million from 2000 to 2015 reaching 264 million registered vehicles. From 2014 to 2015, the transportation demand grew by 3.8 percent and for the past 30 years the traffic congestion level has increased in all urban areas costing the economy in 2014 approximately $160 billion. The average annual delay per commuter rose from 37 hours in 2000 to 42 hours in 2014, an increase of 13.5 percent, i.e. a total delay of 6.9 billion hours, about a third higher than in 2000. The number of highway deaths has increased from 33 thousand in 2010 to 37 thousand in 2016. In the period 2014 to 2015, pedestrian deaths has risen to six thousand, an increase of 21.9 percent. The bureau estimates that people injured in vehicle crashes on highways reached 2.44 million in 2015. Furthermore, the bureau reports that transportation is the second largest producer of greenhouse gas emissions and it accounts for 27 percent of the total U.S. emissions.

These problems have pushed cities into becoming smarter and prompted the development of intelligent transportation systems (ITS) [3] were advanced technologies, such as traffic data collection, traffic data analysis, traffic control and artificial intelligence are applied to improve the overall efficiency of the transportation system.

ITS consists of three essential components, *data collection*, *data analysis* and *information distribution*. This system gathers observable information from the transportation system using various sensors that can measure traffic data such as traffic flow, travel time and road density. They then analyze the collected data to evaluate the traffic conditions and produce traffic control responses. ITSs need to be able to send traffic sensor data to operation centers for evaluation, and thereafter distribute traffic control data from operation centers to commuters and transportation infrastructure [3].

In the past decade, deployment of intelligent transportation systems with features such as real-time traffic monitoring, big data analysis and smart traffic infrastructure has offered unique opportunities for dynamic management of the transportation system and led to improvements in safety and pollution levels.

An important advantage of ITS is its ability to accurately forecast future traffic conditions in a transportation system using enormous amount of traffic sensor data including travel time, traffic density and traffic speed. Traffic predictions allow the system to become proactive in solving problems mitigating them rather than reacting to them. The need for this kind of information is not only crucial for individual commuters and companies for route planning but also for governments when managing the transportation system according to future traffic conditions. This kind of information allows traffic controllers to improve the management of the transportation system by decreasing congestion through rerouting traffic which in turns improves traffic efficiency and safety.

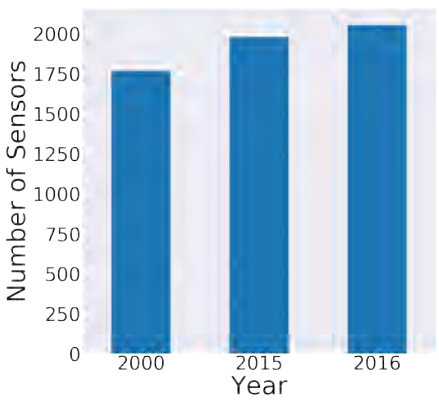
1.1 Problem Definition

Accurately predicting future traffic conditions in real-time is a challenge and not a straightforward task due to the stochastic and nonlinear features of traffic flow [4][5][6]. A widespread deployment of traffic sensors has increased the coverage, quality and availability of

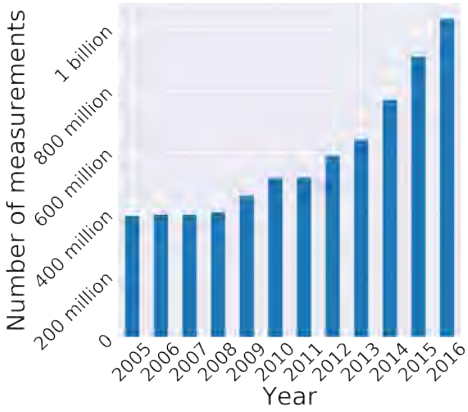
data, prompting a large number of studies of traffic predictions.

In recent years, an increased number of sources for real-time traffic data has emerged from different kinds of infrastructure sensors and mobile data. The success and performance of ITSs have relied on the quality of traffic data. A highly desired feature of ITS is real-time accurate and reliable traffic information. Recently, greater volume and better quality of traffic data has increased the importance of fast, efficient and reliable methods for storing and processing the relatively massive datasets in a scalable manner.

In order to find such an efficient, reliable, and scalable method, traffic data from Trafikverket (*the Swedish Transportation Administration*) covering the cities of Stockholm and Gothenburg will be used in this thesis. Figure 1.1a illustrates how the deployment of traffic sensors has increased from the year 2000 to 2016 in these two cities, while figure 1.1b shows the increase of traffic data measurements from 2005 to 2016 as more of the sensors have come online.



(a) Number of sensors per year



(b) Number of measurements per year

Figure 1.1: Sensor statistics from Stockholm and Gothenburg

1.2 Purpose, Goals and Research Questions

The main goal of the thesis is to explore the feasibility of different ways of producing traffic density predictions by constructing a graph of the transportation system in Stockholm and Gothenburg and partitioning it into subsets of sensors. The graph captures the spatial dependencies between sensors while the sensor measurements are of temporal

nature. This allows for the modeling of both spatial and temporal dependencies.

This approach may provide a more scalable solution to traffic predictions as the number of sensors in the transportation system increases and their traffic measurements become more fine grained, e.g. measurements per minute rather than per hour. However, the approach increases the number of prediction models. The advantage is that these models can be located closer to the physical location of the traffic sensors. Hence, this approach has the potential of producing more frequent and accurate predictions as the latency of transferring traffic measurements to the models from sensors is reduced and the amount of data needed for each prediction is lower.

Figure 1.2 illustrates this hierarchy of data processing to enable a more scalable solution compared to a centralized solution. Data from sensors are sent to a local data processor that produces predictions that can then be sent to either commuters or a central data processor to construct an overview of the predictions for the transportation system as a whole.

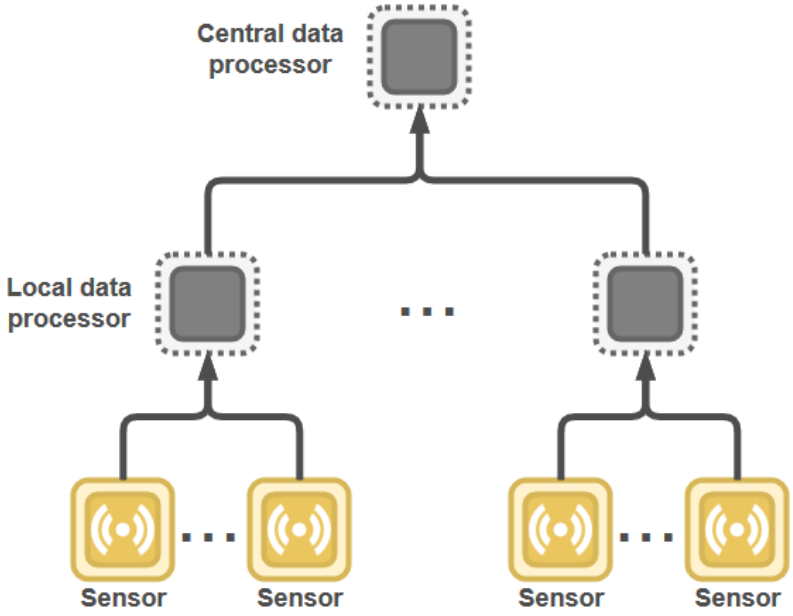


Figure 1.2: Hierarchy of data processing

The thesis will explore and evaluate the prediction error, the training time, and the prediction time of four types of traffic density prediction models:

- Single sensor models where each sensor has its own neural network to predict future density measurements for the sensor.
- A system-wide transportation system model where all sensors in the transportation system are used to train a model to predict future density measurements for all of the sensors.
- Partitioned transportation system models where data from sensors within a partition are used to train models that predict for those sensors.
- Overlapping partition models, where data from a large number of surrounding sensors is used in addition to data from a small number of traffic sensors that the model predicts for.

The research questions of the thesis are:

- Is it possible to construct a graph based structure for traffic sensors in Stockholm and Gothenburg?
- Is it possible to design a graph partitioning algorithm that divides up a graph of sensors based on travel time between sensors?
- Are models created from a partitioned transportation network as accurate in terms of traffic density predictions as single sensor models or a model using all of the transportation network?
- Can the accuracy be improved by generating overlapping partition models that predict for a group of sensors using their traffic flow measurements and measurements from surrounding traffic sensors?

1.3 Methodology

In this thesis, *Triangulation research* method is used as it includes both *Quantitative* and *Qualitative* methods. To understand why these methods are relevant for the study of the thesis, one has to have a clear definition of them and their application in the study.

Qualitative research uses interpretative investigation or development on a sufficiently small dataset to reach reliable theories and conclusions or to develop computer systems, inventions and artifacts [7]. A relatively small dataset of traffic sensors from Trafikverket will be used in this thesis to construct a graph of sensors in Stockholm and Gothenburg. The produced graph will then be used to develop and implement a graph partitioning algorithm that divides the sensor graph based on a user defined criteria for the total travel time between sensors. These partitions will then be used to construct traffic density prediction models.

Quantitative research uses experiments and a large amount of data to reach a conclusion about the validity of a hypothesis or to test a computer systems' functionalities. The precondition for predictability of experiments is that the hypotheses are measurable and quantifiable [7]. The dataset used in this thesis includes one month of traffic measurements from the year 2016 provided by Trafikverket. This data will be applied in the traffic density prediction model experiments and the validity will be verified with a statistical method, i.e. root mean square error.

1.4 Delimitation

The thesis can be divided into three parts. The first part involves analysis of the dataset provided by Trafikverket and the construction of a traffic sensor graph that spans Stockholm and Gothenburg city areas. The graph structure is used to better understand the relationships between the traffic sensors and it is used as the input to the second part of this thesis.

The second part of the thesis is the development and implementation of a graph partitioning algorithm that cuts edges between vertices based on the time it takes to travel between sensor locations. The goal is to form subsets of vertices that are spatially related and conform to travel time requirement specified by the user of the algorithm.

The final part of the thesis is to develop four separate recurrent neural network models that are able to predict future traffic densities. These models are used to evaluate and compare if partitioning a road network of traffic sensors can give comparable results to that of single sensor models and whole transportation system model.

1.5 Contributions

The thesis contributes to the traffic flow prediction research area by investigating and producing the following:

- Graph based structure of all traffic sensors currently deployed by Trafikverket in Sweden, limited to Stockholm and Gothenburg.
- A design and implementation of a sequential weight based graph partitioning algorithm.
- Implementation of four types of neural networks for traffic density predictions, i.e. single sensor, partitioned transportation system, transportation system, and overlapping partition neural networks.
- Evaluation of the prediction error, training time, and prediction time for single sensor models, partitioned transportation system models, transportation system model, and overlapping partition models.

1.6 Ethics

As with any new project developed or applied within the European Union, a fundamental requirement is how the project confirms to the General Data Protection Regulation that came into affect on the 25th of May 2018 Union [8].

The dataset provided by Trafikverket does not contain any personal data or data that can be used to identify individual persons. It only consists of sensor information and aggregate traffic flow measurements from individual sensors. In other words, the measurements from the traffic sensors do not collect information about individual cars but measurements about the traffic conditions per minute at each sensor. This means that individual commuters cannot be tracked or monitored using this dataset as there is no identifiable information.

This thesis does not process the dataset to produce information that can be used to identify commuters. Instead, it seeks to create 1 minute aggregated traffic density predictions.

1.7 Sustainability

The goal of the thesis is to produce traffic density prediction models that are able to make the transportation system more efficient and sustainable. The traffic density predictions may lead to improvements in performance and efficiency of traffic flow as it allows traffic controllers to respond to traffic congestion. Reduction in traffic congestion results in better traffic conditions, fuel savings, and less injuries and fatalities relating to traffic.

By 2030, the Swedish authorities aim to reduce Greenhouse Gas (GHG) emissions from domestic transport by 70 percent compared to 2010 levels. In addition, the Climate Act passed in 2017 requires that Sweden reaches net-zero GHG emissions by 2045 [9]. Traffic is a substantial source of air pollution, statuary noise, and vibrations which causes great deal of health related problems. The Swedish Transport Administration estimates that noise from road and rail traffic causes around 6,700 Disability Adjusted Life Years (DALY) in health losses during one year. In addition, traffic emissions causes health losses on a scale of 27,000 DALY [9]. This thesis may help elevate some of these problems and contribute to the achievement of lower greenhouse gas emission in Sweden.

In 2016, the United Nations Sustainable Development Goals came into effect. This thesis may contribute to solving several of these goals [10]:

Goal 3: Good health and well-being for people. By improving the transportation infrastructure with traffic density predictions, traffic congestion, pollution and accidents can be reduced that in turn promote better health for the general public.

Goal 9: Industry, Innovation, and Infrastructure. The research into scalable and efficient system for traffic density predictions is a step towards improvements and innovation in transportation infrastructure.

11: Sustainable cities and communities. An important part of making cities and communities more sustainable is to improve the transportation infrastructure.

Goal 13: Climate action. The thesis has the potential of contributing to any climate action related to the transportation system of a country.

The most innovative part of the thesis is the effort to create a more scalable and efficient system for traffic predictions which will allow countries to reduce costs on infrastructure, computational resources,

and to increase the speed of predictions to users of the system. Prediction processing units can be located closer to sensors and users of the predictions. This will reduce the need for data to be sent long distances to a central processing unit and allows the use for compression tactics if data is sent to a central location.

1.8 Outline

The thesis is organized into seven chapters. The first chapter addresses the social, economic, and environmental context of the problem definition. The second chapter discusses the theories and concepts necessary to understand the research problem. In the third chapter the focus is on related work. The fourth chapter presents the implementation of the solution developed as a part of this thesis. Experiments are described in detail in chapter five. Results from evaluation of the experiments and implementation are presented in chapter six. Chapter seven discusses the conclusion of the thesis and proposes future work.

Chapter 2

Background

This chapter presents an overview of the main technologies and theories of the thesis. The aim of the chapter is to provide the reader with necessary background knowledge of the thesis research area. This allows future researchers to elaborate further on the findings and contributions of the research presented in this thesis.

The chapter is divided into four sections, i.e. traffic sensors, traffic flow theory, graph theory and machine learning.

2.1 Traffic Sensors

This section describes the most common vehicle detection and surveillance technologies that are used to measure and monitor roads and highways by giving traffic management systems traffic parameters such as vehicle presence, vehicle count and vehicle speed. These technologies are divided into two categorize, i.e. *intrusive* and *non-intrusive* sensors.

Intrusive sensors are placed directly into the road pavement and cause significant disruption to the traffic flow during repairs and installations. The operation of these sensors are well understood and the technology is regarded as both mature and accurate.

Non-intrusive sensors are cost-effective, reliable vehicle detection alternatives that cause minimal disruption of traffic flow during installation and repairs. They are mounted above or to the side of roadways and aim to give at least as accurate measurements as *intrusive* sensors [11].

2.1.1 Inductive Loop Detectors

Inductive Loop Detectors fall under the *intrusive sensor* technology group. Since the early 1960s, these sensors have been used in traffic management systems and are still the most common sensors.

Figure 2.1 depicts the key components of an inductive loop detector. The key components are:

- Insulated wire loop - an insulated wire that is wound in a shallow channel in the road pavement.
- Lead-in cable - cable that runs from the curbside pull box to intersection controller cabinet.
- Electronics unit - housed in the intersection controller cabinet.

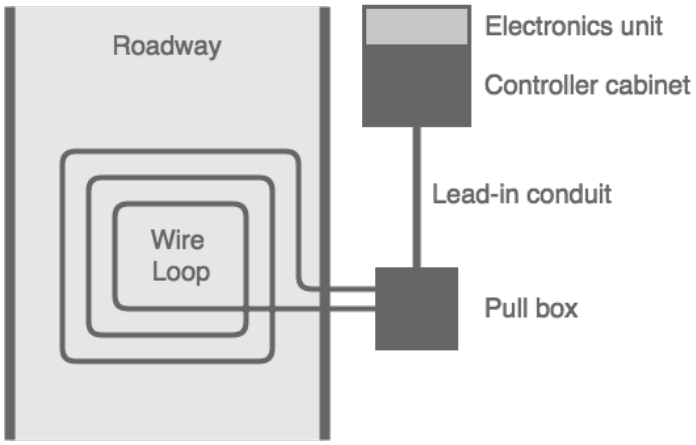


Figure 2.1: Key components of an inductive loop detector

The electronics unit applies electricity to the wire loop and when vehicles drive over or stop within a loop, they cause eddy currents in the wires that decreases the wire inductance. The electronics unit detects the decrease of inductance of the loop and alerts traffic operator of a vehicle presence or passage. Individual loops are not capable of directly measuring vehicle speed. However, the speed can be determined by using either two wire loops spaced apart or an algorithm that calculates the speed by using the loop length, average vehicle length, vehicle detector passage time and number of detected vehicles [12][13].

The advantages of using an inductive loop detector are the maturity of the technology, measurement accuracy and being unaffected by adverse weather conditions. However, installation and repairs are costly both in terms of traffic flow disruptions and pavement cutting [14].

2.1.2 Microwave Radar

Microwave Radars detectors fall under the *non-intrusive sensor* technology group as they can be mounted above or on the side of the road. Modern radar technology was originally developed during the second world war to detect objects such as enemy airplanes. Radar is an abbreviation of the functions it performs, **RA**dio **D**etection **A**nd **R**anging. These sensors can be mounted over the middle of a road lane to measure vehicle data such as volume, speed, occupancy by emitting a beam of energy from its antenna to a limited area of the road and measure the reflected energy as vehicles pass through it.

Figure 2.2 depicts the two types of microwave radar waveforms used by sensors in traffic-related applications.

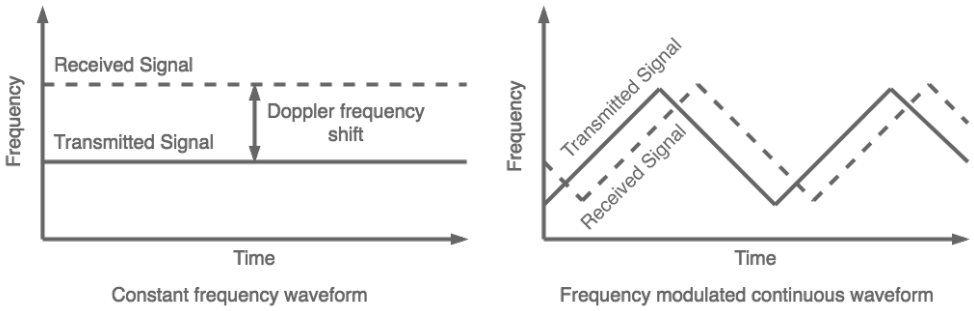


Figure 2.2: The two types of waveforms used in traffic detection

These sensors can emit a continuous wave (CW) or a frequency modulated continuous wave (FMCW). The traffic data received by these sensors is dependent on the shape of the transmitted waveform. CW sensors transmit a signal with a constant frequency in relation to time. By using the Doppler principle, the sensor is able to detect the motion of vehicles. Vehicles moving towards the sensor cause the reflected frequency to shift to a higher frequency. FMCW sensors transmit a signal with a constantly changing frequency with respect to time

and they are therefore able to detect the motion of vehicles and the presence of motionless vehicles [15][16].

Radar sensors are widely used in transportation management systems as they are easy to install, maintain and are unaffected by unfavorable weather conditions [14].

2.1.3 Video Image Processor

Video image processors are in the *non-intrusive sensor* group as they can be mounted outside the road itself. Since the middle of the 1970s, countries including the United States, United Kingdom, Japan, Germany, France and Sweden have been putting parallel effort into investigating video and image processing technologies to be able to replace inductive-loop detectors [17].

The introduction of video cameras in traffic management was initially intended to allow a human operator to interpret a feed of closed circuit television imagery for roadway surveillance. Currently, a more automatic approach is applied, where traffic managers rely on video image processing equipment to analyze and extract important information from the video feed.

Video image processor (VIP) systems usually contain one or more video cameras, a computer and software that processes and analyzes the video feed and converts it into traffic flow data. Fundamentally, a VIP system detects vehicles by analyzing successive frames of a camera video feed that records a traffic scene and calculates traffic flow data based on the analysis of the frames.

The VIP systems that have been developed are categorized into three classes, i.e. *Tripline*, *Closed-loop tracking*, and *Data Associate tracking*.

Tripline allows operators to specify a limited number of detection zones in the video cameras field of view. The system detects vehicles by comparing the changes of the pixels within a detection zone when a vehicle enters the zone compared to the pixels when no vehicle is in the zone. The speed of vehicles is estimated by measuring the time it takes a vehicle to travel a detection zone of a known length.

Closed-loop tracking extends *Tripline* systems by being able to continuously detect and track vehicles along a larger roadway section. The detection and tracking area is limited by the resolution, mounting height and distance of the camera from the roadway. Multiple

vehicle detections are used to validate the vehicle tracking and to improve speed estimates. These systems can provide additional traffic flow information, e.g. vehicle lane-to-lane changes.

Data Associate tracking systems detect and track individual vehicles or groups of vehicles by identifying unique connected areas of pixels. The tracking data for individual vehicle or vehicle group is produced from tracking these connected areas from frame to frame [18][17].

Traffic monitoring with VIP systems gives transportation management systems increased flexibility and lower maintenance cost compared to other systems. However, these systems have been shown to be vulnerable to viewing obstructions, shadows, adverse weather and lighting conditions [14].

2.2 Traffic Flow Theory

Traffic flow theory focuses on understanding and developing an ideal transportation network with negligible congestion and efficient operation by analyzing the interactions between the transportation infrastructure and its users.

Traffic flow theory is primarily used by transportation engineers during the operations stage of a transportation network. The theory is also used during design, construction and maintenance stages. The operational stage deals primarily with analysis and optimization of the operational quality of the transportation network using principles from traffic flow theory.

2.2.1 Performance Measures

Performance measures such as *travel time*, *speed*, *flow* and *density* define the operational quality of the transportation network. Measurements are collected by monitoring the traffic stream that consists of drivers and vehicles as well as their performance and behavioural characteristics.

Flow

Traffic flow theory defines *flow* as the rate at which vehicles travel through a particular point in the transportation network. *Flow* is typi-

cally denoted as vehicles per hour (vph) or vehicles per hour per traffic lane (vphpl).

Capacity and *demand* are two key traffic measures that use the same unit of measurement as *flow* but they cannot be used interchangeably. *Capacity* represents the maximum amount of traffic a road section can handle and *demand* represents the traffic that wants to use a particular road section.

Figure 2.3 illustrates the relationship between flow, demand and capacity. For optimal conditions, the *demand* of a road section is equal to the *flow*. For congested conditions, *flow* becomes equal to the *capacity* of the road section while *demand* is greater than the *flow* [19].

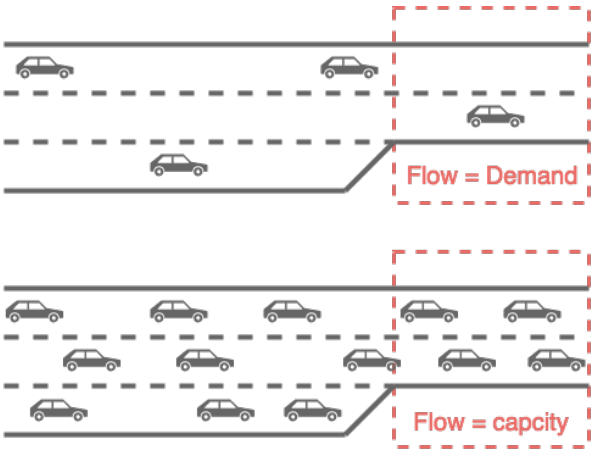


Figure 2.3: Relationship between flow, demand and capacity during non-congested and congested conditions

Speed

Speed can describe both as the movement of one vehicle or the movement of vehicles in a traffic stream. It is measured as distance per unit of time, generally as kilometers per hour (kmph). *Speed* is one of the most useful performance measures as it gives a direct indication if a road section is congested or not. As a direct results of this importance, several speed related terms have been defined for traffic operational analysis and the most frequently used terms are:

Free-flow speed - describes the *speed* of a traffic stream when *flow* is low.

Operating speed - describes the *speed* of a traffic stream during common operating conditions.

Design speed - describes the *speed* for which a road section was designed [20].

Density

Density is a very useful measure for performance as it is defined in terms of units of traffic per unit of distance, generally denoted as vehicles per kilometer (vpkm) or vehicles per kilometer per lane (vpkmpl). With current technology, measuring *density* directly is very difficult. However, mathematically, *density*, *flow* and *speed* are related as shown in equation 2.1. Knowing two of these parameters of the equation allows us to estimate the third parameter [21].

$$D = \frac{F}{v} \tag{2.1}$$

$D = density$ $F = Flow$ $v = speed$

2.3 Graph Theory

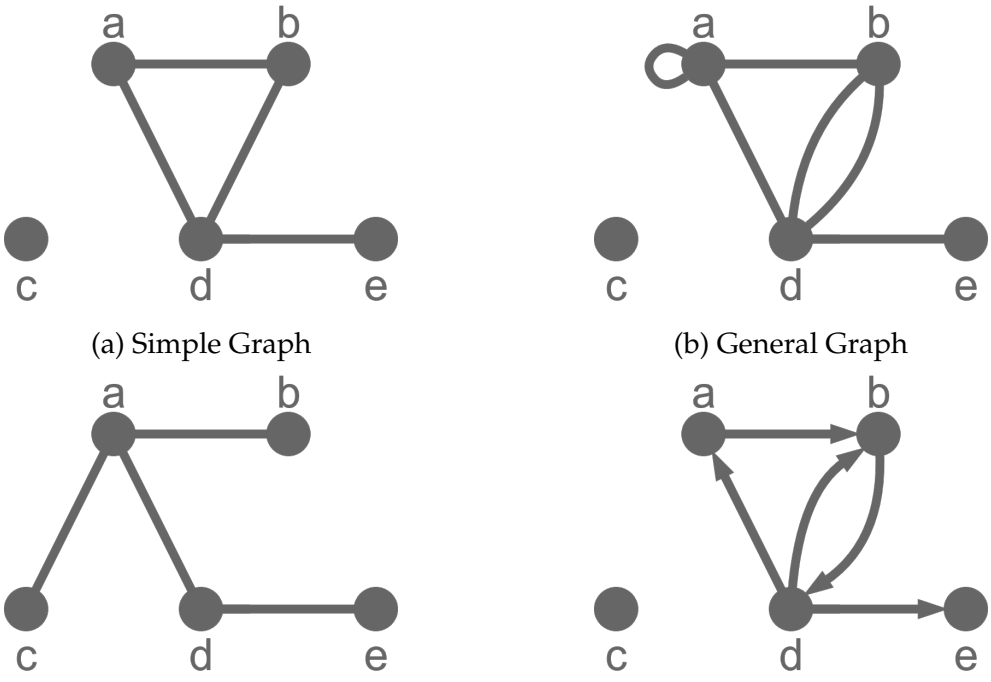
Graph theory is a field within *Discrete Mathematics* and *Computer Science* that studies pairwise relationships between entities using a mathematical structure called *graph*.

2.3.1 Graph Concepts

A *Graph* is a data structure that consists of a set of elements $V = \{v_1, v_2, v_3, ..., v_n\}$ called *vertices* (*nodes*, *points*) and a set of *edges* (*lines*) $E = \{e_1, e_2, e_3, ..., e_m\}$. Each *edge* e_k is an unordered pair of vertices (v_i, v_j).

According to convention, *graphs* are generally represented with diagrams, where *vertices* are depicted as dots and each *edge* is depicted as a line between two dots. Figure 2.4 illustrates four different types of *graphs* using the *vertex* set $\{a, b, c, d, e\}$.

Vertices that have an edge between them are called *neighbors* and are therefore *adjacent* to each other in a *graph*.



(a) Simple Graph (b) General Graph

(c) Connected Graph (d) Directed Graph

Figure 2.4: Graph Examples

A *graph* may contain *loops*, *parallel edges* and *directed edges*. A *loop* is an *edge* that has the same *vertex* for both its endpoints. Figure 2.4b shows a loop for *vertex a*. *Parallel edges* are *edges* that are associated with the same pair of *vertices*. Figure 2.4b depicts two *parallel edges* between *vertices b* and *d*. A *directed edge* is a special type of *edge* where a direction is associated with the edge and the *vertex* pair ordering matters as the *directed edge* (v_i, v_j) represents a direction from the *initial vertex* v_i to *terminal vertex* v_j. Figure 2.4d illustrates *directed edges* between vertices.

Vertex degree ($Deg(v_i)$) is the number of *edges* incident to a *vertex*, where *loops* are counted twice. In figure 2.4b, *vertex a* has a degree of three. *Out degree* of a *vertex* is the number of *directed edges* that have the *vertex* as its *initial vertex* while *in degree* of a *vertex* is the number of *directed edges* that have the *vertex* as its *terminal vertex*. In figure 2.4d, *vertex d* has an *out degree* of three and an *in degree* of one.

A *walk* is a traversal of a *graph*, where each *edge* of a *graph* is only visited once while *vertices* are visited at least once. *Terminal vertices* are the start and the end *vertices* of a *walk*. A *walk* that visits vertices only once is called a *path* and the length of the *path* is the number of edges

it contains.

A *simple graph*, as illustrated in figure 2.4a, is a *graph* that contains *vertices* and *edges* where *loops* and *parallel edges* are not allowed.

A *general graph*, as illustrated in figure 2.4b, is not limited by the same constraints. *Graphs* are allowed to have both *loops* and *parallel edges*. This means that every *simple graph* is a *general graph* but not vice versa.

A *connected graph*, as illustrated in figure 2.4c, is a graph that has at least one *path* between every pair of *vertices*. Otherwise, the *graph* is disconnected as can be seen in figure 2.4b. *Connected subgraphs* within a *disconnected graph* are called *components* of the *disconnected graph*.

A *directed graph* (*Digraph*), as illustrated in figure 2.4d, is a *graph* that contains only *directed edges* and are drawn as arrows that depict the directions a walk can traverse the edge. A *directed graph* is said to be *connected* if the underlying *undirected graph* has at least one *path* between every pair of *vertices* and *strongly connected* if for any two *vertices* there exists a directed path between them [22][23][24].

2.3.2 Graph Separation

A classic computer science problem is how to divide up vertices of a graph into non overlapping groups, clusters or communities. There are two types of algorithms that have been proposed to solve this problem, *graph partitioning* and *community detection* algorithms. The difference between the two algorithms is that in graph partitioning the size and number of the groups is fixed while in community detection they are unspecified.This difference lies in the goals these algorithms are trying to achieve.

The goal of *graph partitioning* algorithms is to find the best division of a graph based on predefined criteria so that each division is small and manageable. These algorithms divide up graphs regardless of whether any good division exists in the graph.

Contrary, the goal of *community detection* is to discover the structure of a graph and the underlying connection patterns that may lie within the graph. These types of algorithms do not divide up a graph if its structure does not indicate a good division [25].

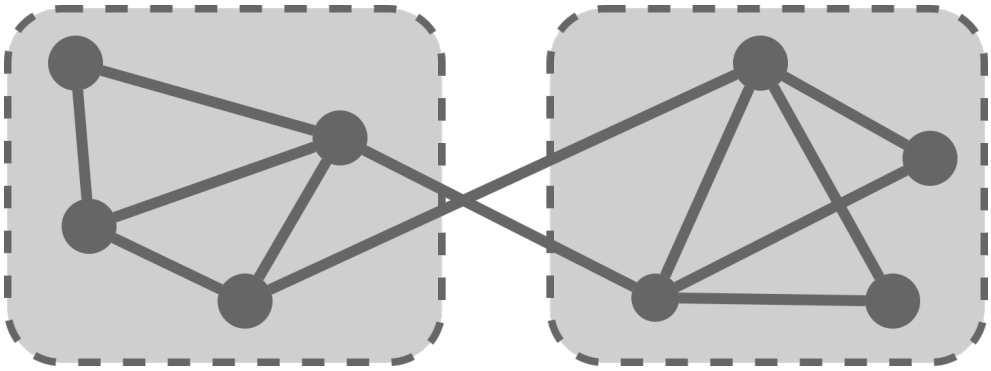


Figure 2.5: Graph divided into two groups of equal sizes

Graph Partitioning

Graph partitioning algorithms aim to divide vertices of a graph into non overlapping groups of fixed size based on specific criteria from the user. This is done to minimize edge cut, i.e. the number of edges between the groups.

Dividing up a graph into two parts, graph bisection, is the simplest graph partitioning problem. The most common approach to arbitrary divisions of a graph is to execute a graph bisection algorithm repeatedly.

Focusing on graph bisection, the simplest solution would be to perform an exhaustive search, try all possible divisions of a graph, and then choose the one with the smallest cut size. The problem with this solution is the excessive computer resource cost of dividing up large graphs.

2.4 Machine Learning

Machine learning is a research field within *artificial intelligence* and has been researched since the early 1980s but has been constrained by the lack of raw training data and computational resources. In the past decade, this has changed with the increased amount of available collected data from various sources and computational resources. Machine learning is a set of tools and methods that allow a machine or a program to extract and identify patterns from the observable world. It gives computers the ability to learn as apposed to being programmed.

Machine learning can be divided into two families, *supervised* and *unsupervised learning*. Supervised learning is when input and desired output information is fed into an algorithm that then finds a way to reproduce the desired output from the input data. Unsupervised learning describes machine learning methods where the output is unknown. Here, the algorithm is fed input data to extract unknown knowledge contained within the data.

In essence, machine learning algorithms extract structural information from raw data and represent it in a structural description called a *model*. The process of creating a model from data is called *training* and the goal of the process is to create an accurate model able to produce accurate output most of the time. These models can then be used to infer things about similar data that has not yet been modeled. They can take many forms such as decision trees representing the data structure as a set of rules or as a parameter vector consisting of neural network weights that represent the connection weights between the neurons.

Datasets used for training models consist of *samples* or data points containing properties called *features* that describe a sample. An example is a grocery store customer dataset consisting of customer samples that have features such as customer name, gender, age and the name of the item purchased. If a model is created to predict what item a customer is going to buy based on age and gender, the machine learning algorithm can be fed with the input features age and gender and the desired output feature of item purchased. Data preparation is an important task of machine learning. An extensive dataset needs to be collected that isn't biased towards any of the desired outputs such that the model becomes as accurate as possible when fed new unseen data. For training of a machine learning model, the dataset needs to be split into two parts. Most of the data, usually around 75%, should be in the *training dataset* while the rest of the data should be in the *test dataset*. The training data is used during the training process of the model while the test dataset is used to evaluated the performance of the model on new data, i.e. data it has not seen before. The performance indicates whether the model is able to *generalize* well, i.e. produce accurate output based on new data.

For the training process, the machine learning algorithm is fed training data to learn from. In the beginning, the model structure is set to some initial values. During the training process, the values are changed iteratively as samples are fed into the model. A *loss func-*

tion is used to calculate how far the produced output is from the desired output. The goal of the training process is to minimize the loss calculated by the loss function for each *training step* by altering the values in the model structure. Once the training process is complete, the produced model is evaluated using the test dataset. This demonstrates how the model may perform on new data and if the training process has produced a good and generalized model. *Overfitting* happens when a model is trained too closely on the training dataset such that it produces accurate output on the training set but not on the test dataset. *Underfitting* happens when the model is too simple and unable to capture the features of the training data producing inaccurate output. *Early stopping* is a condition used to stop the training process before the model overfits the training dataset, i.e. a minimum loss change each training step.

Hyperparameters are parameters used while training models. An example is how often the training process runs through the training dataset (number of epochs) or how large the model structure is. *Hyperparameter tuning* is a process for determining the optimal set of hyperparameters to be used while training a model to improve on the accuracy of the model.

Neural networks are one of the types of machine learning models. A principal unit of a neural network is a *neuron* (*cell*, *unit*) which is based on the biological neuron of the brain. Neurons are connected to allow information to flow between them similarly to biological neurons with synapses. These neurons can transform the information they receive or limit the amount of information they pass along. The inputs of each neuron are often fed into an *activation function* that computes the output of the neuron, e.g. *Sigmoid*, *Tanh* or *ReLU*. The connections between the neurons are called *edges*. These edges usually have a *weight* associated with them that influences the strength of the information sent

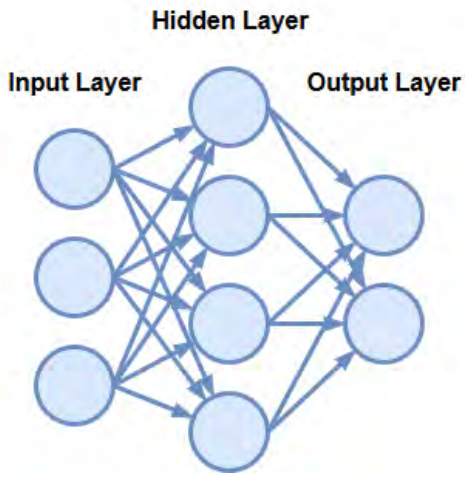


Figure 2.6: Simple neural network

on the edge. The weight is adjusted during the training process. Figure 2.6 illustrates how neurons are organized into layers within a neural network where information flows from the input layer through the network to the output layer, called *feed-forward neural network*. Each layer may perform a specific kind of transformation on their input based on the activation function they have.

Recurrent neural networks extends feed-forward neural networks by including a feedback connection that feeds the output of a hidden layer back into itself. Neurons of the network have feedback loops, allowing information to persist. Figure 2.7 illustrates how a neuron in a recurrent neural network feeds its output with the input value X_t and outputs a value Y_t which is then fed back into itself with the next input value. Recurrent neural networks are exceptionally good for operations over sequences of input vectors where retaining the state of past input vectors is crucial. Tasks where previous data influences the current data, e.g. speech recognition or time-series data. A limitation of recurrent neural networks is that they are unable to learn to connect past information to outputs that are located far into the past. This is a result of the vanishing gradient problem where the edge weights do not update during the training process of a model.

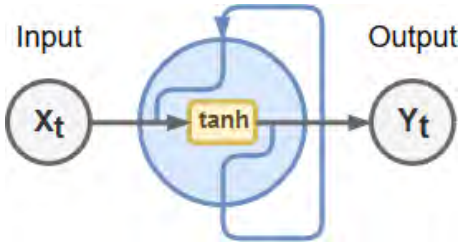


Figure 2.7: Recurrent neural network neuron

To overcome this long-term dependency limitation, *long short-term memory* (LSTM) neuron was developed with gates that allow the neuron to decide which data to keep or forget. Figure 2.8 illustrates the structure of a LSTM neuron.

The green horizontal line is the *cell state* and contains information that the neuron has decided to keep. *Gates* carefully regulate which information is added to the cell state or removed from it. Gates are constructed out of Sigmoid activation functions. They generate a value from zero to one indicating how much information should be let through. Incoming information passes first though a Sigmoid activation function called a *forget gate*. It takes the previous output Y_{t-1} and input value X_t and decides how much of cell state C_{t-1} it should keep. The

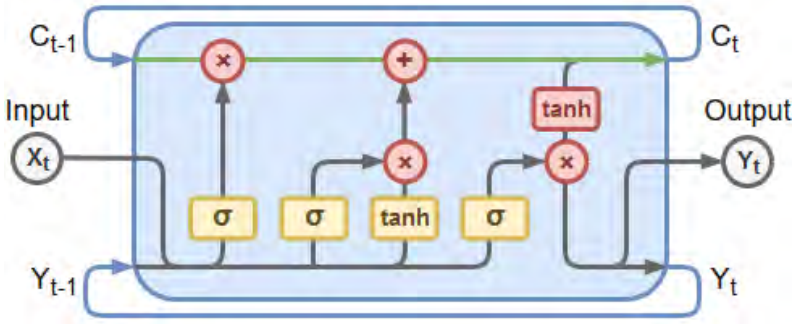


Figure 2.8: Long short-term memory neuron

second gate is a Sigmoid activation function called a *input gate*. It decides which cell state values should be updated from Y_{t-1} and X_t. This gate is combined with a Tanh activation function that produces a vector of candidate values for the new cell state C_t. The last gate is a Sigmoid activation function called a *output gate*. This gate decides from Y_{t-1}, X_t which parts of C_t should be in output Y_t.

For further reading see [26][27][28].

2.5 Traffic Predictions

Approaches for predicting future traffic conditions have been studied extensively for the past decades. They are a critical and important component of intelligent transportation systems and traffic control. Short-term traffic predictions are dependent on complicated multivariate traffic variables and their non-linear interactions.

This section describes three different traffic prediction methods based on the classification proposed by van Hinsbergen, et al. [29] and van Lint, et al. [30]. Figure 2.9 depicts the three overlapping traffic prediction approaches.

Traffic prediction approaches diverge largely from one another in terms of the chosen prediction method, the scale of the predicted transportation system (e.g. single sensor, road or a whole transportation system), and the type of the transportation system (e.g. urban, rural or freeway).

Generally, short-term predictions refer to predictions of up to an hour into the future but vary depending on the traffic prediction approaches. Most studies predict the traffic variables *flow*, *density*, *travel*

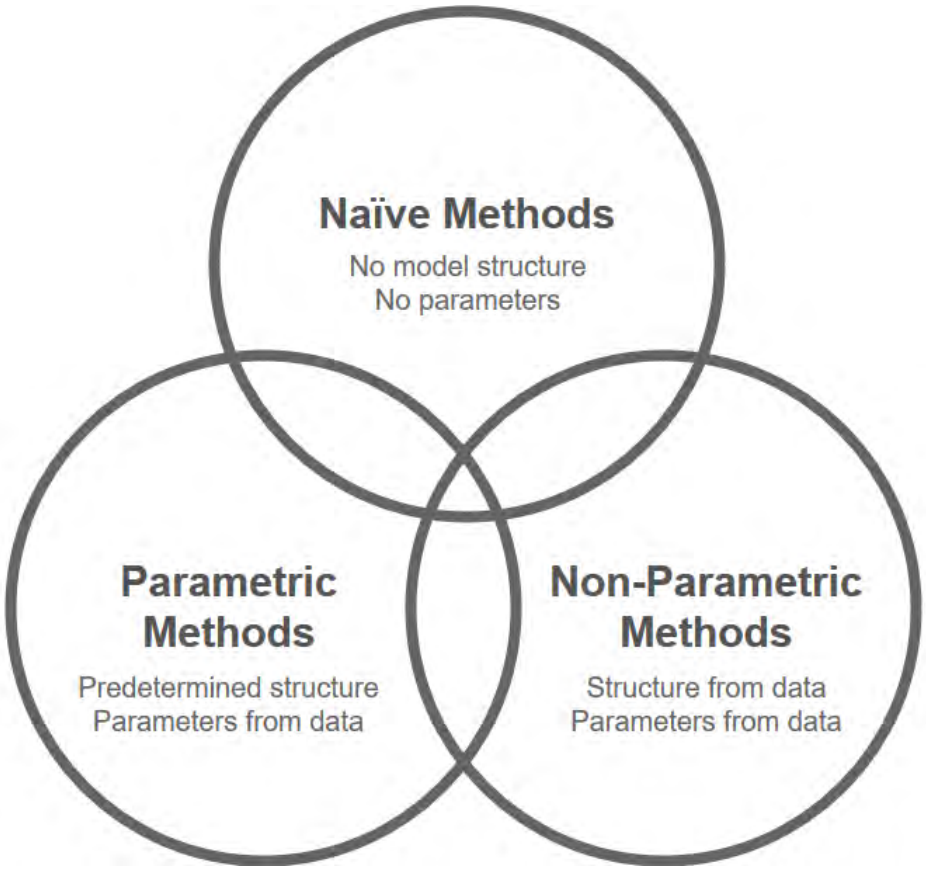


Figure 2.9: Taxonomy of traffic prediction methods

time and *mean speed*. The accuracy of these predictions have either been measured by root mean square error (RMSE) expressing the expected value of the error or by mean absolute percentage error (MAPE) expressing the error as a percentage.

2.5.1 Naïve Methods

Neither model parameters nor model structure are determined from traffic data in naïve methods. Traffic data and its exact physical relationships are used directly such as the relationship between *density*, *flow* and *speed* described in section 2.2 or the usage of the last measured traffic variable as the next future variable.

Because of the low computational requirement and simple implementation, naïve methods are widespread and frequent. However, the

accuracy is typically much lower compared to parametric and non-parametric methods.

Instantaneous Travel Time

Instantaneous travel time assumes that traffic conditions remain constant indefinitely. Previous measured travel time is used as a predictor for the future value. This means that there are no calculations required for this method. Thus, it is extremely fast and easily understandable. High prediction accuracy depends on stationary and homogeneous traffic conditions over long time periods. However, this method tends to have low prediction accuracy as traffic conditions are usually highly dynamic and far from constant [29].

Historical Averages

This method calculates the historical average of a traffic variable from its past values and it uses the results as the predicted future value. These averages are often divided up into periods, e.g. hour of a day or day of the week. This naïve method gives high prediction accuracy for short-term predictions where historical values are distributed around some mean value and have a similar shape throughout the day. However, historical averages are in reality a poor predictor as historical traffic data for most roads are both skewed and have a wide distribution. For long-term predictions, historical averages have shown to give the best predictions accuracy in many circumstances [31].

Clustering

Clustering methods compute groups of historical averages based on discovered traffic patterns present in the traffic data. Algorithms such as *small large ratio* and *ward's clustering* are based on these clustering methods. In some cases, clustering methods are used to preprocess traffic data for non-parametric methods. These clustering methods have been able to outperform methods based on historical averages. They are in some cases more accurate than linear regression models [30].

2.5.2 Parametric Methods

The term parametric signifies that only the model parameters are fitted using traffic data and the model structure is predetermined from concepts in traffic flow theory. These model structures are either analytical, e.g. queuing models or travel time functions, or traffic simulations, e.g. macroscopic or microscopic models.

The advantages of parametric methods are that they are able to model undiscovered incidents, usually need less data than non-parametric methods and have demonstrated high accuracy and good computational performance [29].

Analytical Models

Among the easiest models used by traffic engineers to predict travel time are analytical formulas such as the *bureau of public roads formula* (BPR). The formula describes, on the one hand, the relationship between travel time and traffic flow and, on the other hand queuing functions for travel time predictions in relation to vehicles queuing at a bottleneck.

The main disadvantages of these models are that they require accurate measurements of the input variables, which is rarely the case in reality. Additionally, the parameters and variables have a highly stochastic nature [29].

Simulation Models

Simulation models can be divided into macroscopic and microscopic models.

Macroscopic simulation models consider only global variables such as density, mean speeds and flows of a transportation system. Microscopic simulation models consider only individual vehicles and their interactions within a transportation system [30].

2.5.3 Non-Parametric Methods

Non-parametric methods signifies that both the model parameters and the model structure are not determined in advance but rather from the traffic data itself. For this reason, most traffic prediction models

are categorized as non-parametric, e.g. seasonal ARIMA approaches, recurrent neural networks and nonlinear time-series.

More data is needed for models implemented using non-parametric methods rather than parametric methods as both model parameters and structure are derived from the data. However, the advantage of using these non-parametric methods is that complex, nonlinear and dynamic interactions in traffic can be modeled without the need of domain specific knowledge on underlying traffic processes [29].

Chapter 3

Related Work

This chapter explores previous work in the research field of the thesis. It discusses the advantages of using long short-term memory neural networks as compared to statistical and linear models to predict the stochastic and nonlinear characteristics of traffic flow. The thesis focuses on the scalability and accuracy issues of traffic flow predictions when intelligent transportation systems grow. The goal is to minimize model complexity of traffic prediction models and to allow for fast real-time predictions by means of distributing the prediction models to the edge of transportation system, i.e. close to the physical location of the traffic sensors. This enables a move from a centralized traffic prediction solution to a decentralized one.

Ma, et al. [4] propose a neural network consisting of long short-term memory (LSTM) units to capture the nonlinear features of traffic flow. The authors exploit the memory units capabilities to better capture long temporal dependencies for time series predictions. The study uses 2 minute aggregate travel speed and vehicle volume dataset from two microwave detectors in Beijing to train and test traffic speed prediction models. Their LSTM neural network is composed of an input layer, one recurrent hidden layer and one output layer. Ma, et al. evaluate the proposed LSTM neural network to other recurrent neural networks (Time-delayed NN, Elman NN, and Nonlinear Autoregressive NN), Support Vector Machine regression, Autoregressive Integrated moving Average model, and Kalman Filter approaches. The traffic speed predictions accuracy is measured for each model using different time lags, i.e. 2, 4, 6 and 8 minutes. The study shows that a LSTM neural network is able to outperform the other models with at least 28

percent greater accuracy in all cases except for one.

Polson, et al. [32] developed a deep learning model for traffic flow predictions. They demonstrate that a deep learning architecture is able to capture the nonlinear and spatio-temporal dependencies. The architecture combines a linear model with a sequence of tanh layers. The first layer identifies spatio-temporal relations while the other layers model nonlinear relations. The model trains on a dataset from 2013 with 21 traffic senors located on Chicago's interstate highway I-55. These traffic sensors produce traffic flow measurements every five minutes and the models produce predictions for a traffic sensor located in the middle of the highway. Polson, et al. evaluate the performance of the deep learning architecture with a sparse linear vector auto-regressive model. The study concludes that the deep learning model provides a significant improvement in accuracy over linear models. The authors empirically observe that recent measurements of traffic conditions give stronger indication of future measurements compared to historical data, i.e. last 40 minutes as opposed to last 24 hours of traffic data.

Fu, et al. [6] propose using recurrent neural networks (RNN) with long short term memory (LSTM) and gated recurrent units (GRU) for short-term traffic flow predictions. Models are trained using 50 random traffic sensor from the Caltrans Performance Management System dataset to predict 5 minutes into the future with 30 minutes of historical data. Their experiments show that recurrent neural networks with LSTM and GRU perform better than existing linear models such as the auto regressive integrated moving average (ARIMA) model. The paper concludes that the mean absolute percentage error of GRU RNN is 10 percent lower than ARIMA and 5 percent lower than a LSTM RNN. Additionally, the GRU RNN is able to converge faster than the others.

Dai, et al. [5] propose a deep hierarchical neural network for traffic flow predictions using traffic flow time series data. The neural network is designed to transform traffic flow series into trends describing fixed temporal patterns and residual series for predictions. The neural network consists of two stacked layers, i.e. extraction layer that feeds into a prediction layer. The extraction layer is fully connected and extracts time-variant trends by being feed both traffic flow data

and average trend series. The prediction layer is composed of a long short-term memory units that predict traffic flow using the output of the previous layer and residual series. The study uses a traffic flow dataset collected from 3941 stations that produce measurements every 5 minutes and span California's freeway systems. For training and prediction of the models, the first 16 weeks of 2016 were selected for 50 stations. The architecture is compared in terms of accuracy from 5 minute predictions to traditional models, i.e. *ARIMA*, *MVLR*, *SVR*, *RF*, and deep network LSTM. The study shows that this architecture significantly improves the prediction accuracy.

Cui, et al. [33] propose a deep stacked bidirectional and unidirectional long short-term memory neural network for traffic flow predictions. Forward and backward dependencies in time series data and the spacial features in the data are considered to predict network-wide traffic speeds. Recurring traffic patterns are better predicted using both forward and backward temporal perspectives as upstream and downstream conditions can influence the future traffic conditions. Their architecture increases the depth of the neural network by including a bidirectional layer consisting of two LSTM layers, one for the forward spatial-temporal dependencies and another for the backward spatial-temporal dependencies, that feed into a regular LSTM layer producing the final prediction. To train their models, Cui, et al. use data from 323 traffic detection stations that are a part of the Digital Roadway Interactive Visualization and Evaluation Network (DRIVE Net) system. The models are evaluated by predicting speed 5 minutes into the future using 50 minutes of past data. Cui, et al. compare their model to traditional and state of the art models, i.e. *SVM*, *Random forest*, *feed forward neural network* and a single layer *GRU neural network*. The paper concludes that their deep stacked bidirectional and unidirectional LSTM neural network achieve superior prediction accuracy for the whole transportation system compared to the other models.

Abbas, et al. [34] investigate three long short-term memory neural networks for short term road density prediction. They propose dividing a road network into road stretches and junctions to deal with scalability and accuracy challenges. The paper compares a single sensor (1-1) model using only information from one traffic sensors to predict for the same traffic sensor. A multi sensor (n-n) model utilizing

traffic sensors from a selected area of a highway are considered and predicted for. Finally, a multi sensor (m-n) model is considered using information from few traffic sensors in a selected area to predict for all of the traffic sensors in that area. The study uses fine grained 1 minute traffic flow aggregates from traffic sensors in Stockholm city to predict 10, 20 and 30 minutes into the future. Their experimental results show that using information from neighbouring traffic sensors produces higher accuracy prediction models compared to a single sensor model. It also reveals that traffic sensors located at highway exits give higher prediction accuracy compared to sensors located in middle and at the entrances. Predictions for traffic sensors located at the entrances of the highway have the lowest accuracy. This suggest that the models are able to learn how the traffic propagates through the highways. Abbas, et al. observe that training and prediction times improves when model complexity is reduced by decreasing the input data and memory units. Moreover, the (m-n) multi-sensor model is comparable to the (n-n) multi-sensor model in terms of accuracy. The authors conclude that it is possible to reduce the number of input traffic sensors by 35 percent without compromising the prediction accuracy. This decrease in number of deployed traffic sensors reduces infrastructure costs. Abbas, et al. plan to investigate to what extent prediction accuracy depends on the size of a predicted road segment and whether course-grained aggregation traffic flow measurements are able to increase the prediction accuracy. Finally, they want to examine how to optimally partition a road network in order to minimize the number of deployed traffic sensors within a partition and predict for omitted traffic sensors of the partition.

This thesis builds on insights and results of the research work discussed above. The contribution of the thesis involves experimenting on a larger scale by using traffic flow measurements from all of the traffic sensors in a transportation system in order to achieve a more holistic evaluation of traffic flow predictions. A further contribution of the thesis is automatic partitioning of a transportation system to allow for a scalable and decentralized traffic flow prediction solution.

Chapter 4

Implementation

In this chapter, a detailed overview of the technical design and implementation of the thesis work is presented. The chapter is divided into four sections discussing the data set provided by Trafikverket, the construction of a graph from the datasets, three graph partitioning algorithms, and the design of four different recurrent neural networks for traffic density predictions.

4.1 Datasets

Trafikverket has provided two datasets related to traffic data in Stockholm and Gothenburg. The first dataset contains metadata about traffic sensors that are grouped into traffic sensors' sites located on select highways in these two cities (see figure 4.2). These traffic sensors' sites consist of a group of traffic sensors, which span the lanes of a road at a particular location. Table 4.1 shows the information provided by the dataset. This includes a *McsDsRefer* property containing a road name and number of kilometers from a reference point, which is defined to be at the start of the road in question, *X* and *Y* properties consisting of the GPS coordinates of a traffic sensors' site, and more properties.

Property	**Description**	**Example**
Y	Latitude coordinate	57.6897905867036
X	Longitude coordinate	11.9998409708589
DetectorId	Global traffic sensor identifier	5524
McsDetecto	Traffic sensor identifier at a site	2

McsDsRefer	Road name and kilometer reference of a traffic sensors' site	E6Z 16,060
LaneId	Lane identifier	1
Bearing	Sensor bearing	0
Location	Location description	Kallebäcksmotet E6S 16,060
RegionId	Region identifier	5
StationId	Station identifier	3324
SiteId	Traffic sensors' site identifier	1369
SiteValidF	Date and time traffic sensors' site became valid	2000/01/01 00:00:00.000
SiteValidT	Date and time traffic sensors' site is valid to	9999/12/31 00:00:00.000
DetectorVa	Date and time traffic sensor became valid	2016/05/09 00:00:00.000
Detector_1	Date and time traffic sensor is valid to	9999/12/31 00:00:00.000

Table 4.1: The schema of the metadata dataset

The Second dataset is composed of traffic sensor measurements from radar sensors placed on stationary structures along highways in Stockholm and Gothenburg. These measurements contain traffic data aggregates per minute between the year 2005 and 2016. Table 4.2 describes the measurement schema stored in the dataset.

Property	**Description**	**Example**
Timestamp	Date and time of traffic measurement	2016-11-01 00:01:00.000
Ds_Reference	Road name and kilometer reference of traffic a sensors' site	E75_U 0,905
Detector_Number	Traffic sensor identifier stored in ASCII	50 (2)
Traffic_Direction	Traffic direction stored in ASCII	78 (N)
Flow_In	Number of vehicles per minute	2

Average_Speed	Average speed of vehicles per minute	78
Sign_Aid_Det_Comms	AID, Detector, MSI and Communication raw data	0
Status	Sensor status	3 (OK)
Legend_Group	Sign setup group code	255
Legend_Sign	Sign type	1
Legend_SubSign	Frame type of sign	1
Protocol_Version	Version of MCS Simone/TOP protocol	4

Table 4.2: The schema of the traffic sensor measurement dataset

For this thesis, the measurement dataset needs to be preprocessed to contain density values and filter out noise such as error codes. Data from the month of November 2016 was selected as it had the most complete measurements and reflects best the current transportation system.

The *Average_Speed* property needs to be filtered as values over 251 km/h represents an error code. The value 251 km/h means that the average vehicle speed was less than 2 km/h and is, therefore, converted into 1 km/h. The *Flow_In* property needs to be filtered to only include valid flow measurements, i.e. between 0 and 120 v/h. An unique descriptive traffic sensor identifier needs to be constructed to easily identify traffic sensors using the *Ds_Reference* and *Detector_Number* properties. Density estimates need to be calculated for each sensor per minute using equation 2.1 and the properties *Average_Speed* and *Flow_In*. Any measurements represented with a *NULL* value are replaced with zero (0) as the traffic sensors give an error when there are no vehicles on the road. The 1^{st} and 30^{th} of November need to be removed from the data as these days contain fewer than 1440 measurements per minute or a full day of measurements.

Figure 4.1 illustrates the resulting density measurements from four different traffic sensors in November 2016. This figure shows clearly a recurring pattern as the density rises during the day and declines during the night.

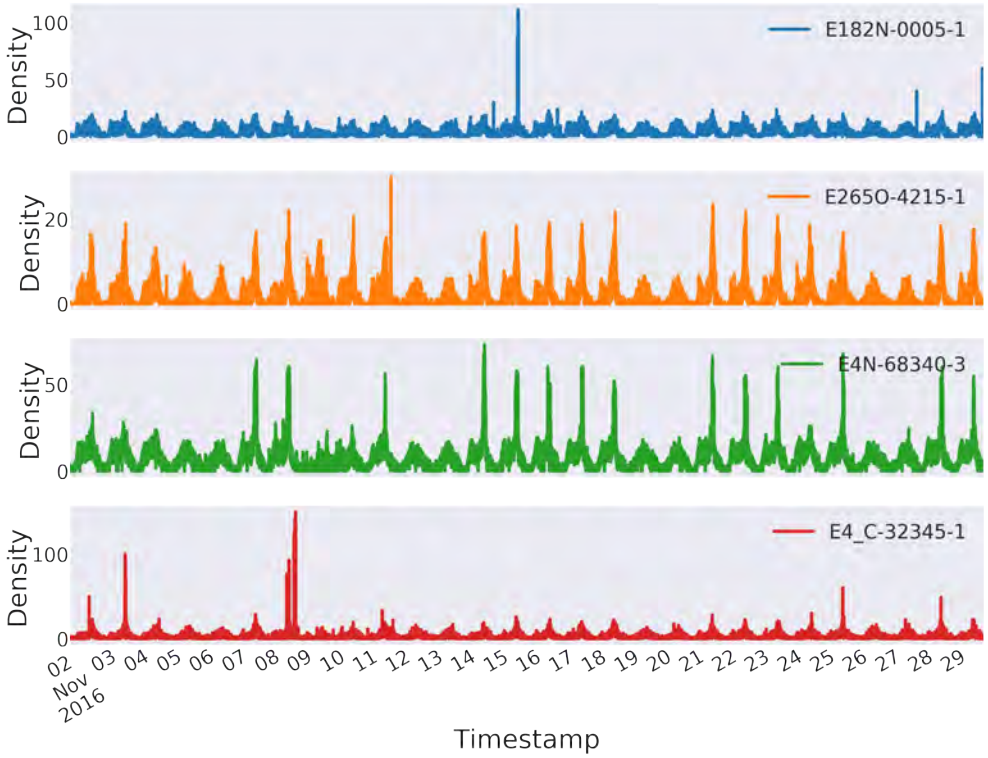


Figure 4.1: Density data from four sensors in November 2016

Figures 4.2a, 4.2b and 4.3 illustrate the traffic sensors' site locations plotted on a geographical map of Stockholm and Gothenburg using the X and Y properties provided with the metadata dataset.

(a) Sensor sites in Stockholm (b) Sensor sites in Gothenburg

Figure 4.2: Traffic sensors' sites in Sweden

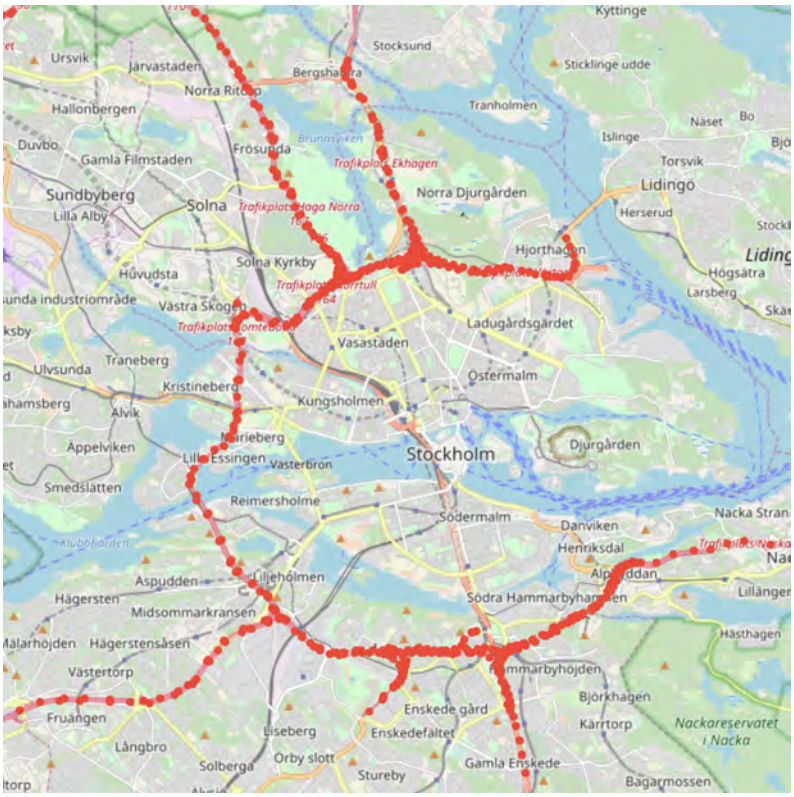


Figure 4.3: Sensor sites in Stockholm center

4.2 Graph

This section describes the construction of a directed weighted graph of traffic sensors' sites representing Trafikverket's intelligent transportation system. As described in section 4.1, the traffic sensor metadata dataset provided by Trafikverket spans both the Gothenburg and Stockholm city areas.

The traffic sensors' site graph consists of vertices which represent physical traffic sensors' site locations for a group of sensors spanning the lanes of the road in question. To describe a vertex of the graph, the *McsDsRefer*, *X* and *Y* properties from the traffic sensor metadata dataset are used. On the basis of the *McsDsRefer* property, it is possible to extract the name of the road and the kilometers from a reference point for a traffic sensors' site. A *Vertex* property is created from the *McsDsRefer* property by replacing white spaces with a hyphen symbol and by removing commas. This new property acts as a unique iden-

tifier for the traffic sensors' site since it is both human readable and helps with manual evaluation of the graph. The *Y* and *X* properties representing latitude and longitude coordinates of a traffic sensor are included in order to plot each site on top of a geographical map. As the aim is to create a traffic sensors' site graph of the current transportation system, the traffic sensors are filtered out based on the *Detector_1* property, which describes the date and time for which the traffic sensor is valid. Hence, only traffic sensors' sites that have traffic sensors currently in use are present in the graph. Two new properties are introduced, *Valid_From* representing the date from which a site is valid and *Valid_To* representing a date to which a site is valid. The values of these new properties are extracted from the properties *DetectorVa* and *Detector_1* of the metadata dataset. Finally, a *Sensors* property is created to represent the number of traffic sensors at a site by counting the number of traffic sensor instances with the same *McsDsRefer* value in the filtered metadata dataset.

Edges in this graph represent physical roads that connect vertices of traffic sensors' sites and the weight of each edge describes the average travel time of a vehicle driving along the road. Extracting the edges from the metadata dataset is simple for traffic sensors' sites located on the same road and can be done in an automatic way. The *Vertex* property can be used to order the traffic sensors' sites based on the road name, the road direction, and the kilometers from a reference point, i.e. a reference point at the start of the road. Edges spanning distances greater or equal to 1 kilometer are removed from the graph, since some roads have traffic sensors' sites that are separated too far apart to be considered as a sequence. These roads are drastically longer than the average road and are therefore not comparable to the others. The reason for this is the possibility of drastic changes in traffic taking place between the two traffic sensors' sites as there may be many entrances and exits on the road. It becomes more complicated to create edges when traffic sensors' sites are connected via a road but have different road names. This means that one cannot rely on the road name and kilometers from a reference point to connect the sites as described above. In this case, edges need to be manually added to the graph. This can be done by displaying the vertices on a map and observing where there are missing edges in relation to the roads.

The next step is to find the weight of each edge. This is done by calculating the average speed of vehicles at each sensor site during

rush hour when most vehicles are on the roads using the *Average_Speed* property, see table 4.2. According to the traffic data measurements from Trafikverket, this occurs between 7:00 to 8:00 and 14:00 to 17:00 o'clock when people typically commute to and from work. Speed during rush hour reflects best traffic behaviour when the density prediction models are most needed. In addition, the time to travel through the Trafikverket's transportation system during free flow is too short for the sequential weight based partitioning algorithm to create many small partitions rather than one big partition. In order to calculate the distance between traffic sensors' sites, one can either use the difference in distance from a reference point if the sites are using the same reference point or the GPS coordinates applying the *Haversine formula*, see equation 4.1.

$$d = 2r \arcsin(\sqrt{\sin(\frac{\varphi_2 - \varphi_1}{2})^2 + \cos(\varphi_1)\cos(\varphi_2)\sin(\frac{\lambda_2 - \lambda_1}{2})^2}) \quad (4.1)$$

$\varphi_1 = start\ latitude\ in\ radians$ $\quad \lambda_1 = start\ longitude\ in\ radians$

$\varphi_2 = end\ latitude\ in\ radians$ $\quad \lambda_2 = end\ longitude\ in\ radians$

$r = Earth\ radius\ (6371\ km)$ $\quad d = distance\ between\ sensor\ sites$

The edge weight is equal to the time it takes a vehicle to travel the road it represents. It is calculated by using the average speed at a traffic sensors' site located at the end of the edge in the direction of the traffic and by using the distance between traffic sensors' sites applying equation 4.2.

$$Travel\ time = \frac{Distance\ between\ sensors(km)}{Average\ speed(km/h)} \quad (4.2)$$

Edges are represented by three new properties, *Source*, *Destination* and *Weight*. *Source* is equal to the *Vertex* property of the traffic sensors' site located at the beginning of the road section in the direction of the traffic. *Destination* is equal to the *Vertex* property of a traffic sensors' site located at the end of the road section. *Weight* is calculated as described above.

4.3 Sequential Weight Based Graph Partitioning

This section presents a comprehensive overview of the design of the three sequential graph partitioning algorithms that have been developed and implemented in this master thesis project.

The reason for designing and implementing a graph partitioning algorithm that is able to partition based on edge weight is to allow for automation of a process that may become overwhelming as the intelligent transportation systems grow. This allows for a quick way to explore how partitioning a transportation system affects prediction accuracy when fed into the design of traffic prediction models. It appears that no one has, so far, designed a graph partitioning algorithm that partitions based on edge weights to produce partitions with a simple longest path of maximum length (sum of edge weights) in relation to a user requirement.

The design of each algorithm is built and based on the fundamental reasoning that vehicles traveling a straight road with one starting point and one end point will pass all the traffic sensors on the road sequentially and in order of the direction of traffic. Following this train of thought, one can presume that the traffic flow values produced from traffic sensors will influence the future traffic flow values produced from the consecutive traffic sensors when vehicles travel the road and pass each traffic sensor. When a vehicle travels at a constant speed on a road with traffic sensors spaced 20 minutes apart, the same traffic flow value will be produced for each of the traffic sensor as the vehicle passes by it.

Figure 4.4 illustrates a simple scenario, where a sensor **s** at a center of a straight road section with one entrance and one exit dependence on the past traffic flow values of the sensors in front and behind it. Vehicles traveling the road section at sensor **s-3** will eventually reach sensor **s** at a future time. Hence, sensor **s-3** influences the future state of sensor **s**. A growing queue or traffic congestion forming at sensor **s+3** will eventually reach sensor **s** given that the congestion will not resolve itself before reaching sensor **s**.

Using the assumption above, three graph partitioning algorithms were developed to divide up Trafikverket's transportation system graph to predict the future traffic flow values for each traffic sensor.

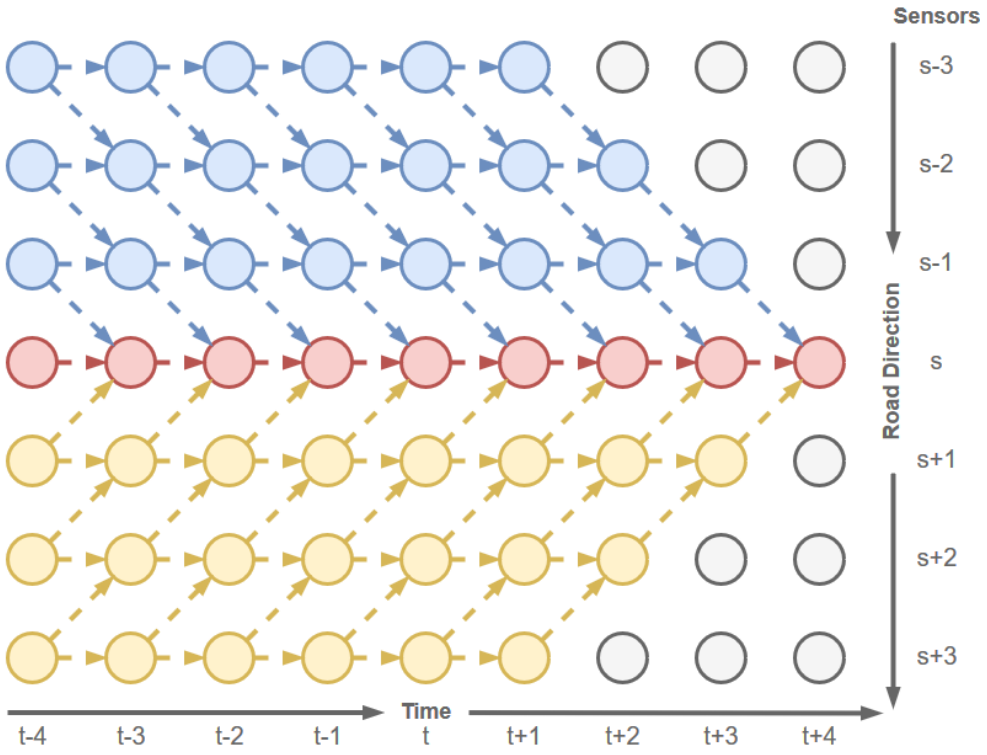


Figure 4.4: A traffic sensor's spatial dependencies

These algorithms depend on a weighted directional graph, where the weights are defined as the time it takes to travel an edge. They also depend on a weight criteria (in minutes) given by the user of the algorithm. The weight criteria defines the upper limit of the total travel time of a vehicle traversing the longest path of a partition. The weight criteria acts as a boundary for each partition while the algorithms partitions a graph. The hypothesis is that the weight criteria represents a prediction time for which a traffic flow prediction model should be the most accurate. Hence, the model contains enough information for a future prediction from traffic sensors such that the sensors at the exits of the partition give an accurate prediction.

The section is split into three subsections that cover the design of each algorithm:

4.3.1 Forward Sequential Weight Based Graph Partitioning

4.3.2 Backward Sequential Weight Based Graph Partitioning

4.3.3 Overlapping Sequential Weight Based Graph Partitioning

4.3.1 Forward Sequential Weight Based Graph Partitioning

The algorithm takes as an input a user specified weight criteria (in minutes) that is used as an upper limit for the total travel time of the longest path a partition has while the algorithm divides up a graph.

The first step performed by the algorithm is to identify a collection of starting vertices that will be used as a starting point from which the algorithm partitions the graph. These starting vertices are characterized by being on the boundaries or edge of the graph or the graph's disconnected components. This means that the vertices do not have any incoming edges from other vertices.

The second step performed by the algorithm is to assign vertices to distinct partitions. This is done by selecting a starting vertex and traversing the graph on the outgoing edges of each vertex it encounters and simultaneously counting the edge weights until the weight criteria is reached. Upon reaching this limit, all the vertices that the algorithm has encountered, so far, will be assigned to a partition. The vertices with incoming edges from vertices within this new partition are assigned to the starting vertices collection and will be used to continue partitioning the graph. This step is repeated until there are no starting vertices left and thereby all vertices have been assigned into a partition.

A special edge case is when the algorithm hits a previously created partition while traversing the graph for a new partition. When the current total weight of the new partition is less than the maximum weight at the connecting vertex in the old partition, they are merged together. If the current weight is greater than the maximum weight of the connecting vertex in the old partition and the weight criteria limit for the partition has not been reached, then the algorithm resets the old partition and adds the vertices at the beginning of the old partition to the starting vertices collection.

Algorithm 1 is a pseudocode of the algorithm described above.

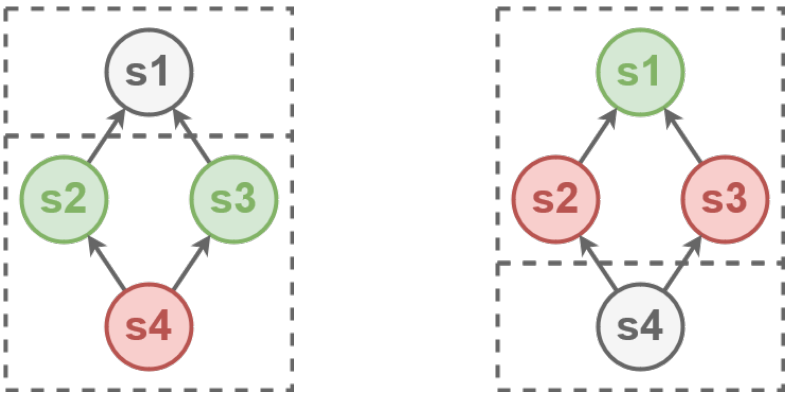
Algorithm 1 Forward Sequential Graph Partitioning Algorithm

```
1: function PARTITION(G, w)    ▷ Where G - graph, c - weight criteria
2:     G.partitions = queue()
3:     G.get_starting_vertices()
4:     while G.starting_vertices not empty do
5:         vertex = G.starting_vertices.dequeue()
6:         G.partition = set()
7:         partition_helper(G, vertex, c, 0)
8:         G.partitions.add(G.partition)
9:     end while
10: end function
11:
12: function PARTITION_HELPER(G, v, c, w)    ▷ Where G - graph, v - vertex, c - weight criteria, w - weight sum
13:     if v in G.partition then
14:         return
15:     end if
16:     if c ≤ w then
17:         G.add_starting_node(v)
18:         return
19:     end if
20:     G.set_vertex_max_weight(v, w)
21:     G.partition.add(v)
22:     for next_vertex in G.forward_vertices(v) do
23:         weight = G.edge_weight(v, next_vertex)
24:         if next_vertex not in G.partitions then
25:             partition_helper(G, next_vertex, c, w + weight)
26:         else if w + weight <= G.get_max_weight(next_vertex) then
27:             p = G.get_partition_by_vertex(next_vertex)
28:             G.partition = merge(G.partition, p)
29:             G.remove_partition(p)
30:         else if w + weight <= c then
31:             p = G.get_partition_by_vertex(next_vertex)
32:             G.add_starting_vertices_from_partition(p)
33:             G.remove_partition(p)
34:             partition_helper(G, next_vertex, c, w + weight)
35:         end if
36:     end for
37: end function
```

4.3.2 Backward Sequential Weight Based Graph Partitioning

This algorithm has the same input parameter (a weight criteria) as the previous algorithm described in subsection 4.3.1. The difference between these two algorithms is the way they select the starting vertices and traverse the graph. Instead of using the vertices in the graph with no incoming edges from other vertices, vertices with no outgoing edges are selected into the starting vertices collection. This algorithm traverses the graph in the opposite direction of the traffic, i.e. traversing the incoming edges of vertices.

The reason for having an algorithm that moves backwards through the graph compared to the direction of traffic is that those vertices located on the exits of the graph become more accurate. These vertices contain more information for a prediction than when using forward partitioning since connecting vertices might be cut off. Figure 4.5 (a) and (b) illustrate how a simple graph might be partitioned differently based on these two approaches. The dotted lines separate each partition. The green sensors should have the best prediction accuracy compared to the other sensors. This is because partitions contain more spatial information for these sensors while the red sensors should have the worst prediction accuracy.



(a) Forward Partitioning (b) Backward Partitioning

Figure 4.5: Difference between forward and backward partitioning

Algorithm 2 contains a pseudocode of the backward sequential weight based graph partitioning algorithm.

Algorithm 2 Backward Sequential Graph Partitioning Algorithm

```
1: function PARTITION(G, w)    ▷ Where G - graph, c - weight criteria
2:     G.partitions = queue()
3:     G.get_starting_vertices()
4:     while G.starting_vertices not empty do
5:         vertex = G.starting_vertices.dequeue()
6:         G.partition = set()
7:         partition_helper(G, vertex, c, 0)
8:         G.partitions.add(G.partition)
9:     end while
10: end function
11:
12: function PARTITION_HELPER(G, v, c, w)    ▷ Where G - graph, v -
    vertex, c - weight criteria, w - weight sum
13:     if v in G.partition then
14:         return
15:     end if
16:     if c ≤ w then
17:         G.add_starting_node(v)
18:         return
19:     end if
20:     G.set_vertex_max_weight(v, w)
21:     G.partition.add(v)
22:     for next_vertex in G.backward_vertices(v) do
23:         weight = G.edge_weight(v, next_vertex)
24:         if next_vertex not in G.partitions then
25:             partition_helper(G, next_vertex, c, w + weight)
26:         else if w + weight <= G.get_max_weight(next_vertex) then
27:             p = G.get_partition_by_vertex(next_vertex)
28:             G.partition = merge(G.partition, p)
29:             G.remove_partition(p)
30:         else if w + weight <= c then
31:             p = G.get_partition_by_vertex(next_vertex)
32:             G.add_starting_vertices_from_partition(p)
33:             G.remove_partition(p)
34:             partition_helper(G, next_vertex, c, w + weight)
35:         end if
36:     end for
37: end function
```

4.3.3 Overlapping Sequential Weight Based Graph Partitioning

To maximize the prediction accuracy of all traffic sensors within a graph, an overlapping graph partitioning algorithm is designed. This is done by creating partitions that overlap and predict for a subset of the traffic sensors within the partition. The overlap ensures that the subset of traffic sensors within the partition have enough information from traffic sensors located in front of and behind the traffic sensor subset. The information from all of the sensors in a partition is used to predict the future traffic flow values of a sensor subset located at the center of the partition. Figure 4.6 shows a simple example of an overlapping partition. Here vertices are grouped into three types: *critical*, *backward* and *forward*. The traffic sensors of type *critical* are the ones that the traffic flow prediction model will predict for. The traffic sensors under the *backward* type provide information about vehicles that might influence the future state of the *critical* sensors as vehicles travel away from them. The traffic sensors in the *forward* type provide information about vehicles in front of the *critical* traffic sensors. An example of information provided are whether vehicle queues are forming due to traffic congestion.

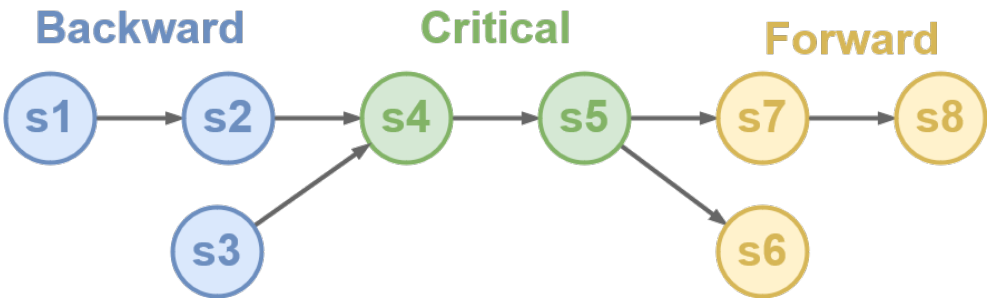


Figure 4.6: Simple example of an overlapping partition

As with the other two algorithms, this one requires a weighted directional graph that will be used to partition. In addition, the algorithm requires the user to specify a *base partition weight*, a *forward overlap* and a *backward overlap*. The *base partition weight* is used as the weight criteria of either the forward or backward partitioning algorithm described above. The resulting partitions define the base partitions from which the overlapping partitions are created. The *forward overlap* indicates how many base partitions located ahead of a *critical*

sensor subset, i.e. in the direction of traffic, should be included in the overlapping partition. It is selected based on how much information ahead of a *critical* traffic sensor subset the traffic flow prediction model needs for a future prediction of the *critical* subset. The *backward overlap* indicates how many base partitions located behind the *critical* sensor subset, i.e. in the direction of traffic, should be included. It is selected based on how much information from traffic sensors located behind a *critical* traffic sensor subset is needed for a future prediction of the *critical* subset. Overlapping partitions are produced by extending each base partition with neighbouring base partitions that form the backward and forward overlap. Within an overlapping partition, the base partition forms the critical subset of the traffic sensors that a traffic flow prediction model will predict for.

The algorithm works in two phases. The first phase uses either of the two partitioning algorithms described in previous subsections using the *base partition weight* as the input *weight criteria* on a weighted directed graph to create the base partitions. Using the generated partitions, a graph of partitions is created where the edges between vertices in different partitions construct a directed edge between the two partitions. The base partitions in this new partition graph form the vertices of the graph and the edges of the vertices in the partition that connect to other partitions from the edges of the new partition graph. The second phase of the algorithm generates overlapping partitions from the new partition graph, where each base partition is extended to include partitions ahead of it, based on the specified *forward overlap* and partitions behind it, based on the specified *backward overlap*.

Algorithm 3 Overlapping Sequential Weight Based Graph Partitioning

```
1: function PARTITION_WITH_OVERLAP(G, w, f, b)                ▷
   Where G - graph, w - base partition weight, f - forward overlap, b
   - backward overlap
2:     partition_graph(G, w)
3:     PG = generate_partition_graph(G)
4:     for p in PG.partitions do
5:         for next_p in G.forward_partitions(p) do
6:             overlap_helper(PG, p, next_p, 0, f, True)
7:         end for
8:         for next_p in G.backward_partitions(p) do
9:             overlap_helper(PG, p, next_p, 0, b, False)
10:        end for
11:    end for
12: end function
13:
14: function GENERATE_PARTITION_GRAPH(G)      ▷ Where G - graph
    that has been partitioned
15:    PG = new PartitionGraph()
16:    for partition in G.partitions do
17:        f_partitions = G.get_forward_partitions(partition)
18:        b_partitions = G.get_backward_partitions(partition)
19:        PG.add(partition, f_partitions, b_partitions )
20:    end for
21:    return PG
22: end function
23:
24: function OVERLAP_HELPER(PG, p, next_p, c, m, f)           ▷
    Where PG - partition graph, p - partition, next_p - next partition, c
    - current overlap, m - max overlap, f - is it forward overlap
25:    if c ≥ m then
26:        return
27:    end if
28:    PG.add_overlap_vertices(p, next_p)
29:    for n_p in G.next_partitions(next_p, f) do
30:        overlap_helper(PG, p, n_p, c + 1, m, f)
31:    end for
32: end function
```

4.4 Neural Networks for Traffic Density Predictions

As has been discussed in chapter 1, real-time traffic predictions in intelligent transportation systems are of great importance as it allows the system to become proactive rather than reactive to adverse traffic conditions. Proactive intelligent transportation system increases the overall efficiency of the system and is thereby of immense value to modern society.

Traffic flow data is captured in the form of time series where each traffic sensor produces traffic flow measurements in time order. The data exhibit distinct spatial and temporal dependencies as vehicles drive along highways from sensor to sensor and traffic flow fluctuates depending on the time of the day. These patterns of spatial and temporal dependencies can be separated from other traffic fluctuations and can be modeled to make predictions. For many decades, the stochastic and nonlinear nature of traffic flow has remained a challenging problem for achieving accurate predictions. The coverage, amount and granularity of data has increased enormously and catapulted research in the field due to the wide deployment of traffic sensors [4].

Traditional methods for traffic flow predictions has mainly focused on statistical and linear models such as autoregressive integrated moving average (ARIMA) that are unable to express the nonlinear and stochastic characteristics of traffic flow. Traffic flow predictions using neural networks (NNs) have proven to be able to outperform these statistical models when predicting time series data. They are able to better deal with the high dimensional data and modeling the nonlinear relationships [4].

Recurrent neural networks (RNNs) are especially suitable to capture the dynamic, temporal and spatial nature of the transportation system as they utilize memory units that allow them to evaluate incoming time series data using what they have learned about past conditions of the transportation system. Even though traditional RNNs show greater ability to model nonlinear time series data, a major drawback is the vanishing gradient problem causing RNNs to fail to capture long dependencies. To solve the vanishing gradient problem, special memory units, long short-term memory (LSTM), have been designed containing memory blocks that are able to selectively store or forget in-

formation. Achieving accurate long-term dependencies is particularly important for traffic flow predictions and LSTM has been designed for time series data [4].

A deep multilayer LSTM architecture has been found to be more accurate than simple RNNs to predict complex nonlinear relations in traffic flow data. These simple neural networks are too shallow to adequately capture the spatial and temporal dependencies compared to deeper networks. This architecture stacks multiple neural networks such as LSTM on top of each other, where each preceding LSTM hidden layer feeds its output to the next LSTM layer as its input [34].

Models produced from deeper neural networks are progressively better at representing the data, but at the cost of increased complexity. Additionally, the growing amount of data produced by deploying more traffic sensors increases the model complexity as the number of model parameters grow. This may lead to unfeasible increase in computational resource requirements, training time and prediction time. Especially, when prediction models need to wait for a traffic measurement from the slowest traffic sensor. A relatively small city like Stockholm is no exception as the number of traffic sensors deployment increases when the Intelligent Transportation System grows.

The thesis proposes and compares four types of LSTM recurrent neural networks for traffic density predictions based on the partitioning algorithms introduced in section 4.3. The aim is to solve the issues of scalability as the number of traffic sensors, model complexity, and prediction time grows, while taking into account spatial and temporal dependencies. Each type of neural network represents a different level of partitioning granularity, i.e. from a transportation system as a whole to a single traffic sensor.

All of the proposed neural networks are based on the same architecture consisting of a multi-layer recurrent neural network that enables them to effectively learn the stochastic and nonlinear characteristic of traffic flow within a transportation system. This architecture is based on the findings in the research paper [34].

Each neural network consists of six layers stacked on top of each other. Each layer feeds its output into the next layer as its input. The first layer is an input layer of n traffic sensors with p past traffic density values for each traffic sensor. The last layer is an output layer of m traffic sensors with f future traffic density values for each traffic sensor. The first layer is followed by two LSTM layers that contain L LSTM

hidden units each. These layers feed into a densely connected layer with 500 hidden units that feeds into another densely connected layer with $m \times f$ hidden units that feeds into the output layer.

If the transportation system consists of $S = \{s_1, s_2, ..., s_n\}$ traffic sensors, the neural network consumes time-series data of past values $p = \{t_{-9}, t_{-8}, ..., t_{-1}, t\}$ for each traffic sensors in S where one t contains one minute average of traffic density values measured by a traffic sensor. The neural network predicts the future time-series density values of $f = \{t_{+1}, t_{+2}, ..., t_{+30}\}$ for each of the traffic sensors in $Z = \{s_1, s_2, ..., s_m\}$ where $Z \subseteq S$. The number of traffic sensors in the input and output layers follow the rule: $n \geq m \geq 1$. The past density values *p* contain nine past density values and one current density value, i.e. 10 minutes of past density values. The future density values *f* contain thirty future density values, i.e. 30 minutes of future density values. All of the neural networks presented in this thesis have an input of 10 minutes of past traffic density values and predict 30 minutes of future traffic density values. The results of performing a hyperparameter tuning procedure on each of the neural networks determines the number of LSTM hidden units selected, *L*.

4.4.1 Transportation System Neural Network

The simplest form of partitioning is to treat the whole transportation system as one big partition. Figure 4.7 illustrates the structure of this *n to n* neural network. Here, traffic density values from the whole transportation system are used to predict for the whole transportation system. This means that the neural network can learn spatial dependencies between all of the sensors and it has greater overview of the whole transportation system. The produced traffic density prediction model is the most complex one, as *n* is equal to the number of traffic sensors in the whole transportation system, i.e. 2034 traffic sensors. This requires that the neural network has the largest number of LSTM hidden units to able to learn traffic flow for the whole system. Of the four proposed neural networks, this one produces the most complex model. 1000 LSTM hidden units proved to give the most accurate predictions after hyperparameter tuning the neural network, as will be discussed further in section 6.3.1. Furthermore, the speed of producing real time predictions is dependent on the slowest sending traffic sensor since the model produced depends on the traffic density values

from all traffic sensors.

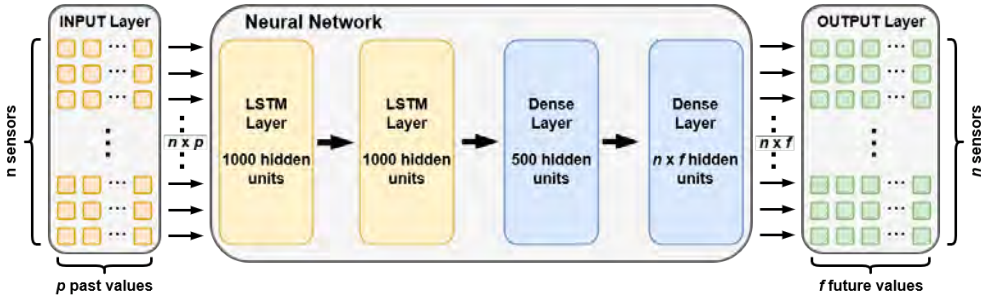


Figure 4.7: Transportation System Neural Network

4.4.2 Partitioned Transportation System Neural Network

The next level of partitioning is to partition the transportation system into smaller sections that are spatially related. This method exploits the spatial dependencies between traffic sensors by taking neighbouring sensors and reducing the number of needed LSTM hidden units. This provides a more scalable approach to traffic flow predictions while giving accurate predictions. The partitioning algorithms discussed in section 4.3 are used with the partitioning weights 3, 5, 10 and 20 to produce the partitioned transportation system neural networks.

Figure 4.8 illustrates the structure of these neural networks, where n is equal to the number of traffic sensors in each partition. The L number of LSTM hidden units is selected based on hyperparameter tuning one random partitioned transportation system neural network for each of the partitions produced by each of the partitioning weights, see section 6.3.1.

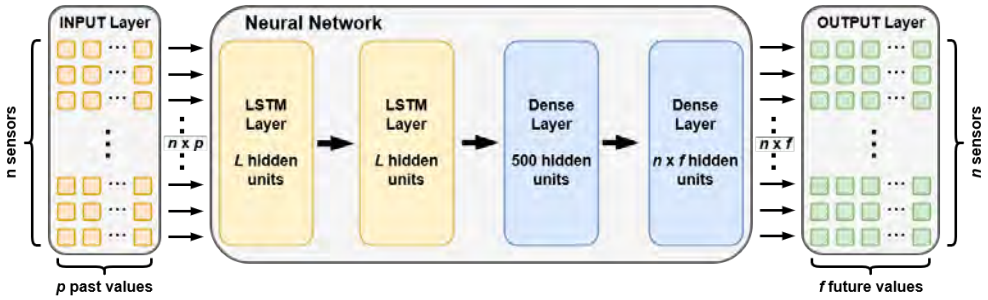


Figure 4.8: Partitioned Transportation System Neural Network

Table 4.3 shows the different number of LSTM hidden units used in each version of the partitioned transportation system neural network. It also contains the result of hyperparameter tuning the neural networks as discussed in section 6.3.1.

Partitioning minutes	Number of LSTM hidden units
20 minutes	1000
10 minutes	800
5 minutes	600
3 minutes	200

Table 4.3: Number of LSTM hidden units

4.4.3 Single Sensor Neural Network

A single sensor neural network is an extreme form of partitioning, where each sensor is its own partition. Similar to the above neural networks, this neural network takes the past time series traffic density values from one traffic sensor and predicts its future values.

Figure 4.9 illustrates the structure of the *1 to 1* neural network, where the neural network only considers information from one traffic sensor without information from the neighbouring traffic sensors. As less time-series data is needed for each traffic density prediction only 50 LSTM hidden units are used. This has proven to give the most accurate results when performing hyperparameter tuning as discussed in section 6.3.1. The produced traffic density prediction model is less complex compared to the models produced from the above neural networks.

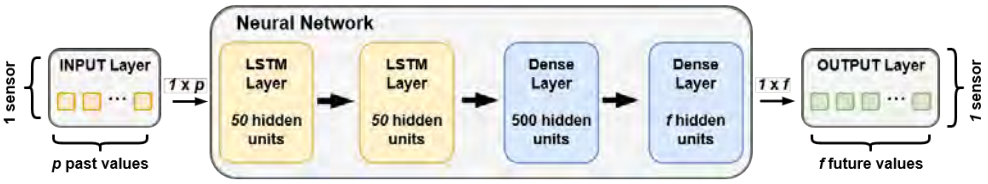


Figure 4.9: Single Sensor Neural Network

4.4.4 Overlapping Partition Neural Network

The final proposed neural network is an overlapping partition neural network that was designed in order to increase the prediction accuracy for all of the traffic sensors in Trafikverket's transportation system. The neural network predicts for a small section of the transportation system by using the traffic flow data from a larger encapsulating section. This allows the produced model to use the spatial dependencies of many neighbouring traffic sensors to predict the future state of the system for a few traffic sensors.

Figure 4.10 illustrates this *m to n* neural network structure. Using the results from performing hyperparameter tuning on the partitioned transportation system neural networks with partitioning weights 10 and 20 minutes, the number of LSTM hidden units selected is 1000. The results are applicable because the overlapping neural network is using 2 minute forward partitions and 10 minute backward partitions to predict a 3 minute partition such that in total 15 minutes of data is used for each overlapping partition. Here $m \geq n$ depending on the overlapping partitions produced.

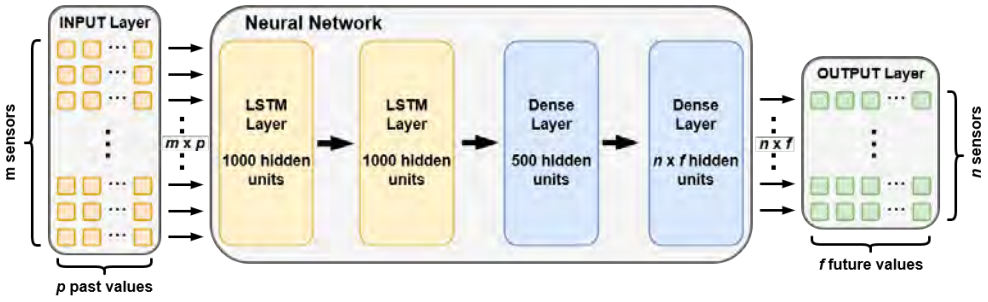


Figure 4.10: Overlapping Partition Neural Network

Chapter 5

Experiments

This chapter describes the experiments and evaluation of the constructed graph, graph partitioning algorithms and traffic density prediction models described in chapter 4.

5.1 Graph

Traffic sensor metadata dataset provided by Trafikverket is presumed to be accurate in terms of the physical location of the traffic sensors in relation to the provided GPS coordinates and acts as a ground truth for the construction of the graph. This is due to the fact that Trafikverket is responsible for the installation and maintenance of the traffic sensors described in the dataset they provide. The constructed graph should be accurate in terms of the location of the traffic sensors' sites in relation to the location of the other sites in the graph. Edges in the graph should correspond to a physical road connecting two traffic sensors' sites and the edge weight is the time it takes to drive the road.

As a means to verify the correctness of the graph, all vertices and edges are plotted on top of a geographical map and the Google Maps Street View technology [1] is used to navigate visually the roads. Both of these methods allow verification of the vertex location in relation to its corresponding traffic sensors' site and connecting edges to the roads between the traffic sensors' sites. When drawing the graph on top of a geographical map, the vertices located at road exists are colored in orange, and vertices located at road entrances are colored in green, while

[1] https://www.google.com/streetview/

other vertices are in blue color. Edges are depicted as red lines with the end of the edge drawn thicker than its start to depict the direction of traffic. Each edge is then manually verified by observing the roads connecting the traffic sensors' sites and their edge counterpart using both methods described above.

Figure 5.1 illustrates how most edges and vertices look on top of a map, where we have edges running straight along highways with entrances and exists. The correctness of this type of road section is relatively easy to verify in this way.

Figure 5.1: An example of a straight road section in Stockholm

Figure 5.2 shows a complicated road interchange in central Stockholm, where the manual verification approach is more difficult. In this case, edges seem to intersect due to the fact that the corresponding roads are located in tunnels on top of each other. Most of the graph is constructed automatically using the road name and kilometers from a reference point for a traffic sensors' site. Sites with the same road name indicate that the sites are located on the same road. This allows

the focus to be on verification of a few edges that represents roads connecting traffic sensors' sites located on roads with different names as they cannot be automatically created.

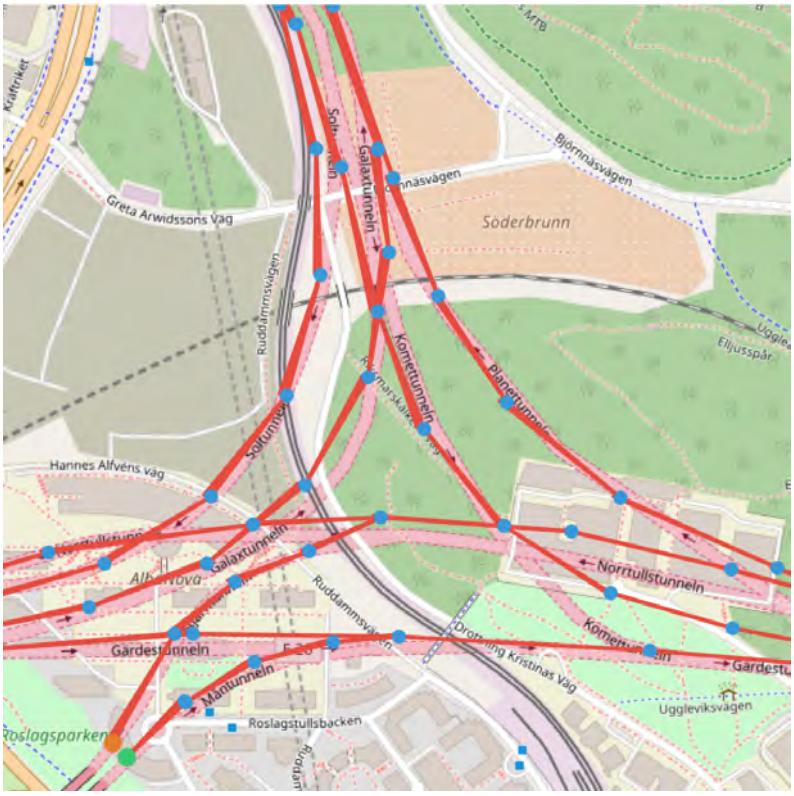


Figure 5.2: An example of a complicated road interchange in Stockholm

Instead of verifying the weight of each edge indicating the travel time distance measure of a road, the equations described in section 4.2 are used.

5.2 Sequential Weight Based Graph Partitioning

The criteria for partitioning correctly a weighted directional graph is that the partitions produced from the algorithm correspond to a weight criteria specified by the user of the algorithm. This weight criteria acts

as the upper limit for the total travel time of the longest path a partition has, i.e. the maximum length of the simple longest path (sum of edge weights). Most importantly, the partition must be a connected component when the partition is converted to an undirected graph, i.e. there is a path between any two vertices.

Twelve partitioning scenarios were designed in order to verify the correctness of the proposed partitioning algorithms from section 4.3. Each scenario contains vertices that represent sensors in a theoretical transportation system, where each arrow refers to a road and each associate arrow number indicates the time it takes to travel the road. Each color represents a partition that the algorithm is anticipated to create from the transportation system graph. The numbers placed outside of a vertex depicts the current weight that the algorithm has accumulated while traversing the graph.

Figure 5.3 illustrates one of the simpler scenarios designed, where the graph has only one starting vertex *s1* as it is the only vertex that does not have any incoming edges. The scenario requires the algorithm to split the partitions based on a weight criteria of value 3 resulting in two partitions, i.e. vertices marked with green and vertices marked with red.

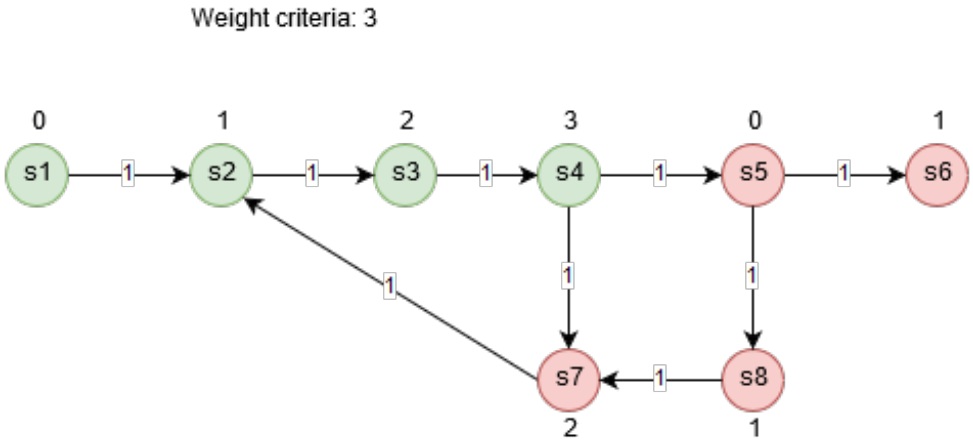


Figure 5.3: Simple evaluation scenario

Figure 5.4 demonstrates a more complex scenario, where the end result depends on the starting vertex. This scenario requires the algorithm to partition a graph with a weight criteria of value 3. There are two possible starting vertices with no incoming edges. In both cases, two partitions are created but they are quite different from each other.

The above graph uses the starting vertex *s1* while the below graph uses the starting vertex *s9*. These two results are, however, equally correct.

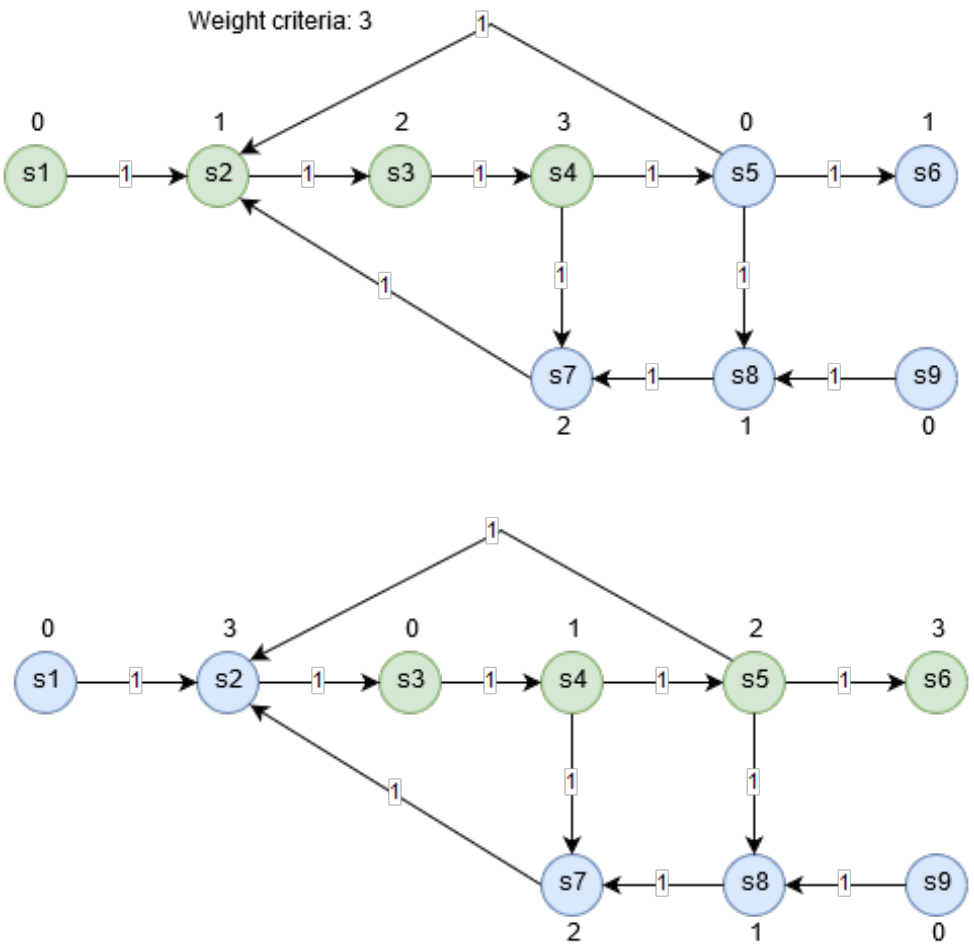


Figure 5.4: Complex evaluation scenario

Figure 5.5 illustrates a scenario to verify the correctness of producing overlapping partitions using the proposed algorithm in subsection 4.3.3. In this scenario, the algorithm partitions the theoretical transportation system graph using a base partition weight of value 1, a forward overlap of value 1, and a backward overlap of value 1.

Three partitions are created by backward partitioning the graph. Each partition is then extended by combining one partition ahead and one partition behind resulting in three overlapping partitions.

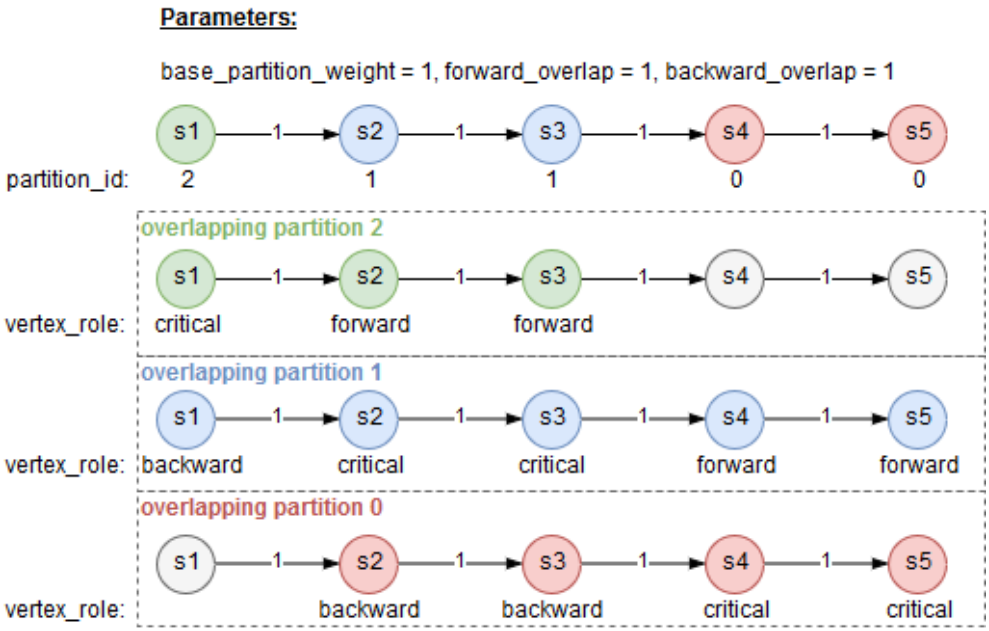


Figure 5.5: Overlapping evaluation scenario

5.3 Traffic Density Predictions

The traffic density prediction experiments conducted as a part of this thesis aim to evaluate the feasibility of partitioning a transportation system for traffic density predictions. The experiments explore the difference in prediction accuracy and execution time to determine if partitioning is a viable way to scale traffic flow predictions as transportation systems grow.

Real traffic data collected from traffic sensors installed and maintained by Trafikverket covering both Stockholm and Gothenburg city areas are used in the experiments. This data is applied when evaluating the four proposed neural networks, see further description of the traffic flow data in section 4.1. The traffic flow data is from November 2016 consisting of traffic flow measurements collected every minute from 2034 traffic sensors located in the cities of Stockholm and Gothenburg. Density values from each traffic sensor have been calculated for each of the traffic sensor during November 2016. They are then nor-

malized and scaled to be in the range 0 to 1 using equation 5.1.

$$d\_scaled = \frac{d - d\_min}{d\_max - d\_min} \times (scale\_max - scale\_min) + scale\_min \tag{5.1}$$

$$d = input\ density \qquad d\_scaled = normalized\ density$$
$$scale\_max = 1 \qquad d\_max = Maximum\ density$$
$$scale\_min = 0 \qquad d\_min = Minimum\ density$$

The resulting dataset is split into 30% testing data and 70% training data. The training data is further split into 20% validation data and 80% training data. The entire transportation system was trained and tested with the four proposed neural networks using the last 10 minutes of density values from each traffic sensor to predict the next 30 minutes of traffic density values.

All experiments are performed using a computing cluster provided by the HopsWorks platform[2]. Each experiment was configured to use 4 executers with 4 virtual cores and 16GB of RAM running Spark 2.3.0 and TensorFlow 1.8.0 using the Keras API implementation. The traffic density prediction models are trained using the Adam optimization, which is an extension of stochastic gradient descent. It maintains a per-parameter learning rate that is adapted using the first and second average moments of the gradients to improve performance. The evaluation metric mean squared error (MSE) is used as the loss function and is minimized by the model, see equation 5.2. Models are trained using data batches of size 100 to calculate the loss. A maximum of 100 epochs is used, which is the maximum number of times the model is trained on the whole training dataset. An early stopping function is implemented that monitors the validation loss and stops the training procedure when the minimum change in MSE is less than 0.0001.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{5.2}$$

$$Y_i = actual\ density \qquad \hat{Y}_i = predicted\ density$$
$$n = number\ of\ traffic sensors$$

[2]https://www.hops.io/

To evaluate the accuracy of the trained models, each of them was used to predict the future density values of the traffic sensors based on the test dataset. The root mean squared error (RMSE) was calculate to find the difference between the actual density values and the predicted density values. The training time of each proposed neural network was measured while being trained for all of the traffic sensors in the transportation system as well as the prediction time of the trained models.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{5.3}$$

$Y_i = actual\ density$ $\hat{Y}_i = predicted\ density$

$n = number\ of\ traffic sensors$

Further evaluation of the models is made by grouping traffic sensors into three groups based on the their location within a partition and then comparing the results with the other models. First group represent traffic sensors located at the entrances of a partition. The second group represents traffic sensors located at the exits of a partition. The third group of traffic sensors is located between the other two groups. The RMSE value for all these three groups is calculated and compared to the same traffic sensors in the other traffic density prediction models.

5.3.1 Hyperparameter Tuning

Hyperparameter tuning is performed on each proposed neural network to establish the best number of LSTM hidden units to be used by each. This is done with *Grid Search* where a set of parameters are specified and tested, see table 5.1. For the single sensor neural network a random traffic sensor was selected and a model trained with each of the hyperparameters. The validation loss of each is then evaluated to select the best performing hyperparameter.

Neural Network	LSTM units
Whole Transportation System	50, 100, 500, 1000, 1500, 2000
20 minute Partitions	100, 200, 400, 600, 800, 1000
10 minute Partitions	100, 200, 400, 600, 800, 1000
5 minute Partitions	100, 200, 400, 600, 800, 1000
3 minute Partitions	100, 200, 400, 600, 800, 1000
Single Sensor	50, 100, 500, 1000

Table 5.1: Number of LSTM units selected for hyperparameter tuning

Chapter 6

Results

This chapter presents the results from the implementation introduced in chapter 4 and the experiments conducted in chapter 5. There are three sections in this chapter describing the graph, sequential weight based graph partitioning and traffic density predictions.

6.1 Graph

Two datasets have been created on the basis of the traffic sensors' site graph construction. The first dataset describes vertices of the graph representing traffic sensors' sites. A site contains a group of traffic sensors spanning the lanes of the road, see table 6.1.

Property	**Description**	**Example**
Vertex	Traffic sensors' site identifier	E18_E-29765
McsDsRefer	Road name and kilometer reference of a traffic sensors' site	E18_E 29,765
Y	Latitude coordinate	59.3882214026729
X	Longitude coordinate	17.9392604786576
Sensors	Number of sensors at a site	2
Valid_From	Date and time traffic sensors' site became valid	2001/01/01 00:00:00.000
Valid_To	Date and time traffic sensors' site is valid to	9999/12/31 00:00:00.000

Table 6.1: The schema of the vertex dataset

The second dataset contains information about the edges of the graph. Table 6.2 demonstrates the schema describing the edges representing roads that connect two traffic sensors' sites. Source indicates the traffic sensors' site at the beginning of a road connecting two sites. Destination represents the traffic sensors' site at the end of a road. Weight contains the travel time of a vehicle driving the road.

Property	**Description**	**Example**
Source	Identifier of a traffic sensors' site at the beginning of a road section	E20_A-59100
Destination	Identifier of a traffic sensors' site at the end of a road section	E20_A-59225
Weight	Travel time distance in minutes	0.20493029927405

Table 6.2: The schema of the edge dataset

Figure 6.1 shows Trafikverket's intelligent transportation system that is stored in the two datasets discussed above. The diagram is drawn using the Kamada-Kawai path-length cost-function provided in the NetworkX Python library.

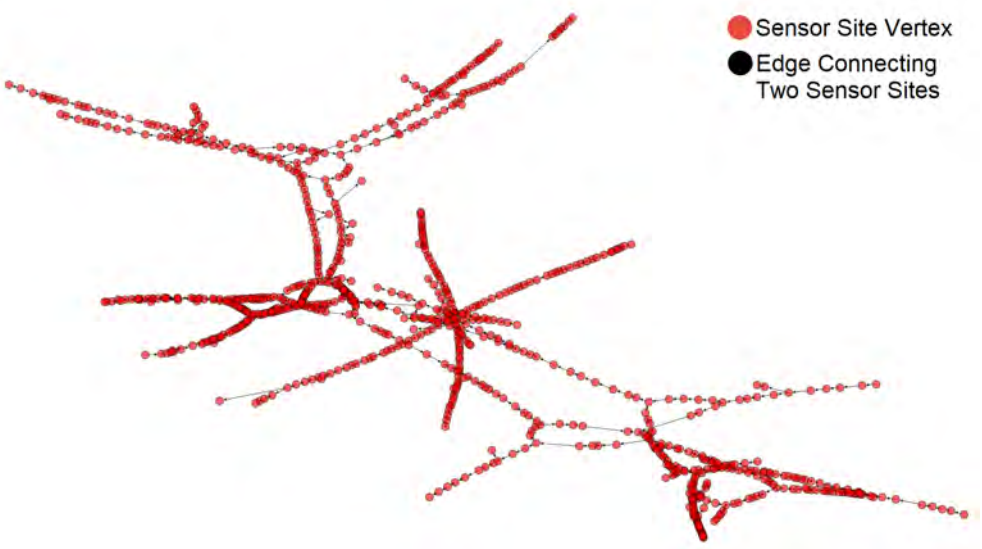


Figure 6.1: The directed graph of Trafikverket's traffic sensors' sites

These datasets can also be used to plot the whole traffic sensor system on top of a geographical map. This is demonstrated in figures 6.2

for Stockholm and in 6.3 for Gothenburg as well as in figure 6.4 for Stockholm center.

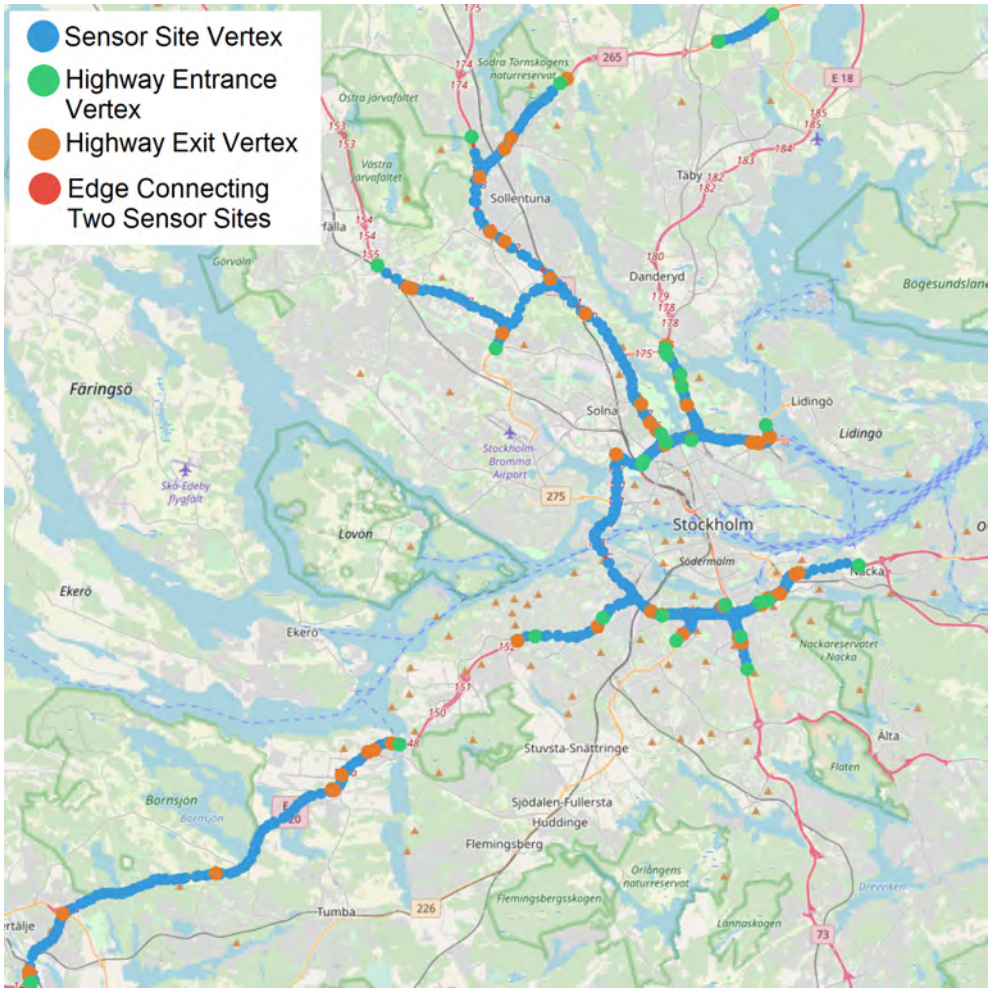


Figure 6.2: Trafikverket's traffic sensors' site graph of Stockholm overlaid on top of a map

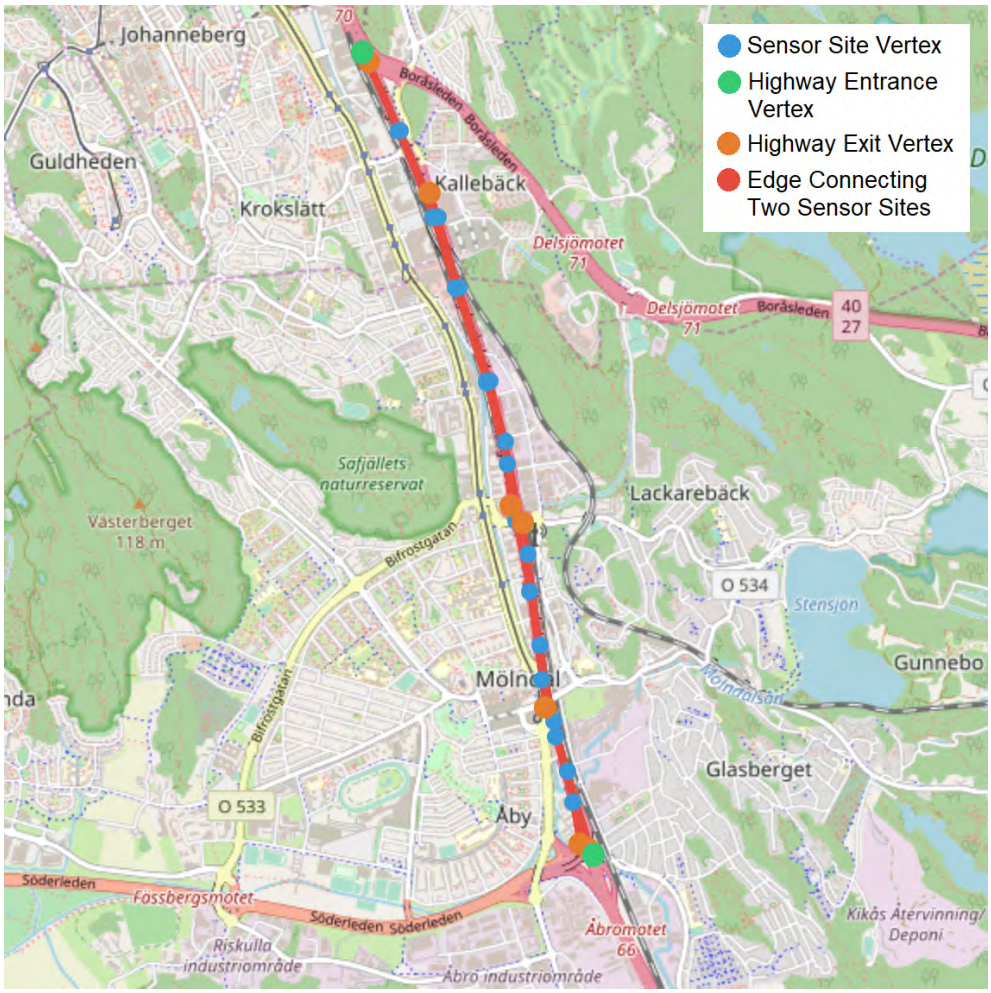


Figure 6.3: Trafikverket's traffic sensors' site graph of Gothenburg overlaid on top of a map

Figure 6.4: Trafikverket's traffic sensor site graph overlaid of Stockholm center on top of a map

Table 6.3 shows statistical information about the constructed traffic sensors' site graph. There are seven disconnected components within the graph. The disconnected components can be identified in the figures above by the long stretches of road where there are no vertices or edges separating sections of the highways. As the table 6.3 shows, there are 857 traffic sensors' sites, 2034 traffic sensors and 2.37 traffic sensors on average at each site in the graph. The average travel time of an edge is 13.45 seconds. This is because each traffic sensors' site is only separated by a few hundred meters.

Number of vertices	857
Number of edges	868
Number of traffic sensors	2034
Number of disconnected components	7
Number of entrance sites	33
Number of exit sites	59
Average weight of an edge	13.45 seconds
Maximum weight of an edge	1 minute 36.75 seconds
Minimum weight of an edge	2.02 seconds
Average number of sensors at a site	2.37
Maximum number of sensors at a site	6
Minimum number of sensors at a site	1

Table 6.3: Graph statistics

6.2 Sequential Weight Based Graph Partitioning

The transportation system graph created in section 6.1 is partitioned with the backward sequential weight based graph partitioning algorithm. Figures 6.5 to 6.8 illustrate how the graph was partitioned with the weight criteria of 3, 5, 10, 20 and 30 minutes travel time. As can be seen from the difference between the figures, the traffic sensors located in Stockholm city are partitioned differently based on the weight criteria provided. It is possible to observe in the figures that the traffic sensors within a partition are continuous and the partitions are separate from each other. Some traffic sensors within a partition seem to form groups detached far apart from each other. This is due to limited amount of available colors to represent all the different separate partitions. Hence, the identified groups are in fact separate partitions. Some colors appear to be intermingled with each other when two roads with traffic in opposite direction are located side by side and are in separate partitions.

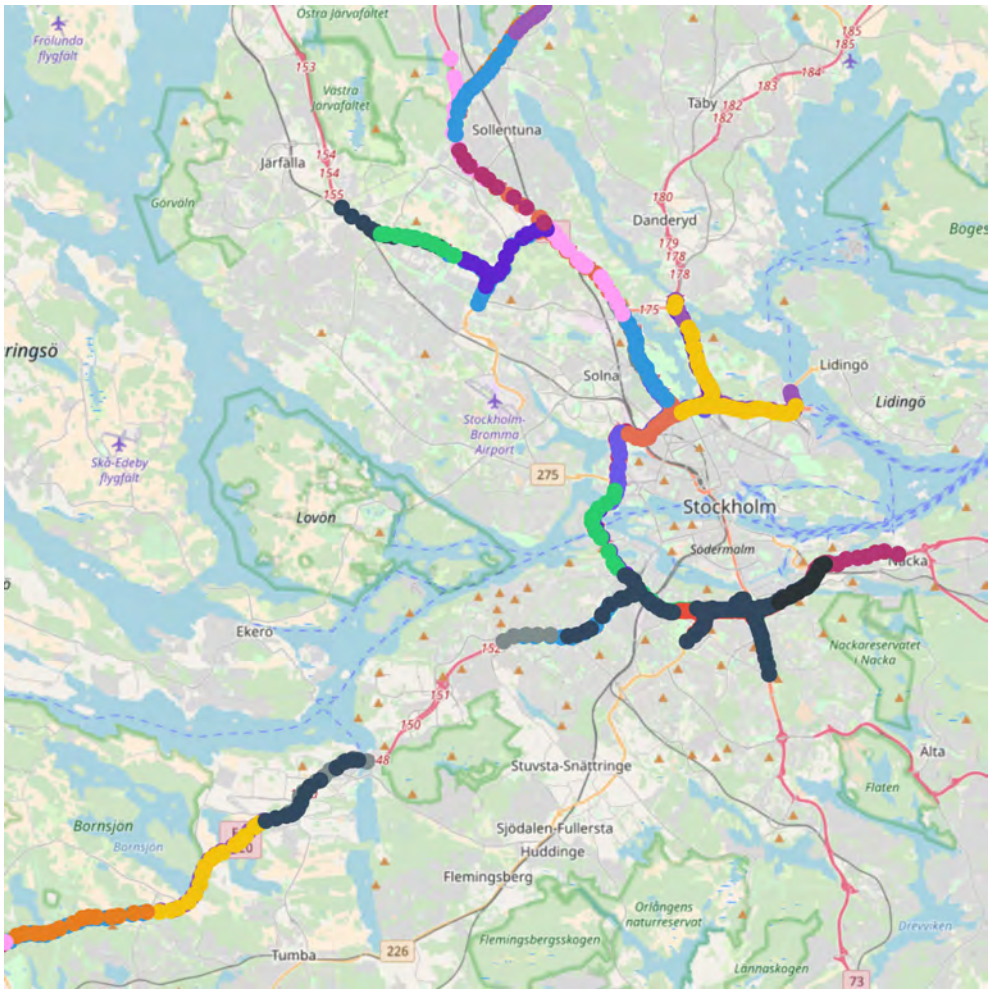


Figure 6.5: Results for Stockholm when partitioning the transportation system with a weight criteria of 3 minutes

Figure 6.6: Results for Stockholm when partitioning the transportation system with a weight criteria of 5 minutes

Figure 6.7: Results for Stockholm when partitioning the transportation system with a weight criteria of 10 minutes

Figure 6.8: Results for Stockholm when partitioning the transportation system with a weight criteria of 20 and 30 minutes

Table 6.4 contains general statistics about the generated partitions. As can be seen from the information in the table, the number of partitions decreases as the weight criteria is increased. In addition, the number of vertices within each partition can vary significantly. Finally, the minimum number of vertices in a partition is one vertex which suggests either that a traffic sensor site is located far away from the rest or that there is a special case not accounted for by the algorithm present in the graph. The table clearly shows that partitioning with a weight criteria of 20 minutes and 30 minutes produces the same partitions. Hence, the transportation system only needs to be partitioned up to the weight criteria of 20 minutes. This results in traffic prediction models based on 3, 5, 10 and 20 minutes partitions.

Weight Criteria	3 min	5 min	10 min	20 min	30 min
Number of partitions	36	19	13	7	7
Maximum number of vertices in a partition	194	245	408	678	678
Minimum number of vertices in a partition	1	1	1	11	11
Average number of vertices in a partition	23.8	45.1	65.9	122.4	122.4

Table 6.4: Partitioning statistics

Figure 6.9 illustrates a partition from the overlapping graph partitioning algorithm with a *base partition weight criteria* of 2 minutes and 3 *forward partitions* as well as 10 *backward partitions*. The blue sensors' sites are a part of the *forward* type. The green are a part of the *backward* type and the red are a part of the *critical* sensors' site type.

Figure 6.9: Example of a overlapping partition in Stockholm

Table 6.5 shows the statistics from generating overlapping partitions using a weight criteria of 2 minutes, adding 3 forward partitions and 10 backward partitions.

Base weight (Weight criteria)	2 min
Forward overlap	3 partitions
Backward overlap	10 partitions
Number of partitions	62
Maximum number of vertices in a partition	609
Minimum number of vertices in a partition	11
Average number of vertices in a partition	268.9

Table 6.5: Overlapping partitioning statistics

6.3 Traffic Density Predictions

This section discusses the results from the traffic density prediction model experiments. The experiments evaluate the feasibility of partitioning a transportation system to increase the scalability of traffic flow predictions for a growing transportation system.

Table 6.6 shows the number of experiments conducted as a part of this thesis. In total, 2224 experiments were carried out using the produced traffic density prediction models from the proposed neural

networks. The number of experiments for each model is determined by the number of partitions produced from the graph partitioning algorithms when using the constructed transportation system graph.

Experiment	Number of executions
Single Sensor Model	1938
3 Minute Partition Models	72
5 Minute Partition Models	38
10 Minute Partition Models	26
20 Minute Partition Models	14
Transportation System Model	2
Overlapping Partition Models	124
Hyperparameter Tuning	8
Total	2224

Table 6.6: Overview of the experiments conducted

Table 6.7 contains the results of the experiments conducted. It shows how the training time, prediction time and LSTM units grow as more traffic sensors are predicted for. Partitioning produces smaller models that are faster both in training and prediction time. The prediction accuracy of the models measured by the average RMSE of all traffic sensor predictions has a standard deviation that ranges only from 0.06 to 0.2.

The results of the experiments show that partitioning a transportation system is an effective method to scale traffic predictions of a growing transportation system. The proposed neural networks produce traffic prediction models that are able to give comparable or in some cases better prediction accuracy than a whole transportation system model and single sensor models.

This is because the neural networks are designed around spatially related traffic sensors such that the models produced are only fed with information directly impacting the traffic sensors being predicted for. Less complex models are produced since less data is fed into these models allowing for fewer LSTM hidden units resulting in faster predictions. This allows the distribution of the models to be closer to the physical location of the traffic sensors that reduces the amount of time each prediction requires as predictions depend on the slowest producing traffic sensor.

Model	Avg Training Time	Avg Prediction Time	LSTM Units	Avg # of Sensors	Avg Input Size	t+3	t+5	t+10	t+20	t+30
Single Sensor Models	145.475536	28.806837	50	1	10	3.739942	3.919132	4.227438	4.670886	5.005468
3 Minute Partition Models	213.637071	5.484408	200	53.833333	538.333333	3.917492	4.051177	4.303766	4.714600	4.989993
5 Minute Partition Models	1062.938760	21.500250	600	102.000000	1020.000000	3.972693	4.090591	4.304761	4.664080	4.923701
10 Minute Partition Models	2298.757915	58.234825	800	149.076923	1490.769231	4.135995	4.223154	4.407602	4.715076	4.893071
20 Minute Partition Models	3493.143713	92.653748	1000	276.857143	2768.571429	4.267915	4.339546	4.502577	4.750380	4.909344
Transportation System Model	6478.674998	132.962821	1000	1938	19380	4.345503	4.398094	4.707944	4.727141	4.861865
Overlapping Partition Models	4017.255924	106.947988	1000	608.661290	6086.612903	3.858814	3.990396	4.227652	4.581103	4.806628
Standard Deviation						0.205859	0.166773	0.160901	0.052302	0.064291

Table 6.7: Overview of the experimental results

Figure 6.10 uses the training time and average traffic sensor numbers from table 6.7. It illustrate how the training time grows as more traffic sensors are introduced into the system while the prediction accuracy remains the same. In addition, the figure shows how partitioning is able to reduce the training time while retaining the same level of accuracy.

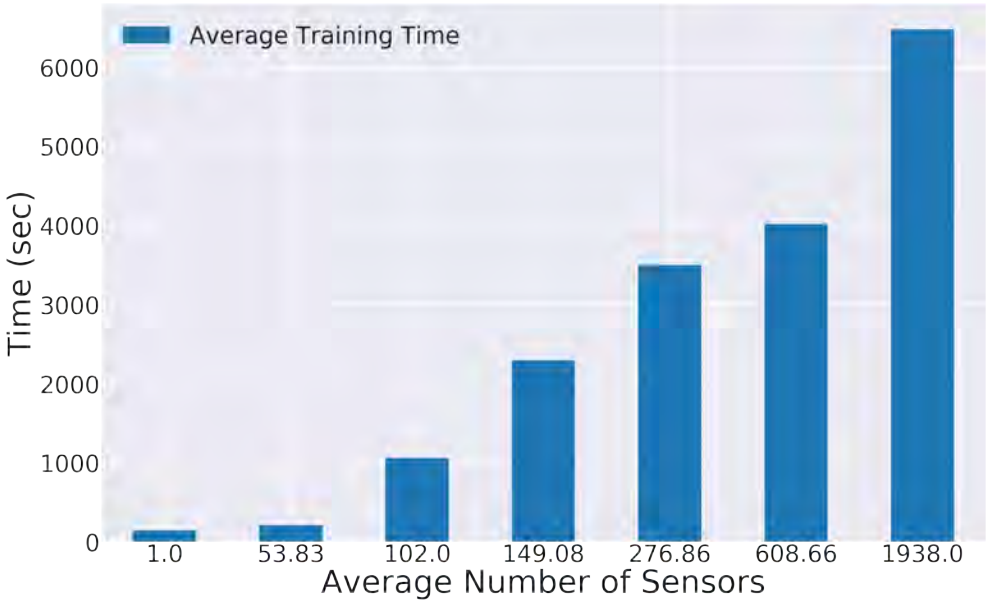


Figure 6.10: Relationship between training time and average number of traffic sensors

Similar to figure 6.10, figure 6.11 uses the prediction time and average traffic sensor numbers from table 6.7. It also shows that prediction time increases with greater average number of traffic sensors while retaining the same level of accuracy. The higher prediction time of a single traffic sensor is due to the environment in which the experiments were carried out. HopsWorks provides a compute cluster for many researchers and companies. Hence, time measurements may be affected by load from other ongoing projects. The single sensor models were trained sequentially and their training took longer than the other types of models. They are, therefore, more susceptible to adverse load on the cluster. However, the results give a clear estimate of the training time and prediction time for each model type.

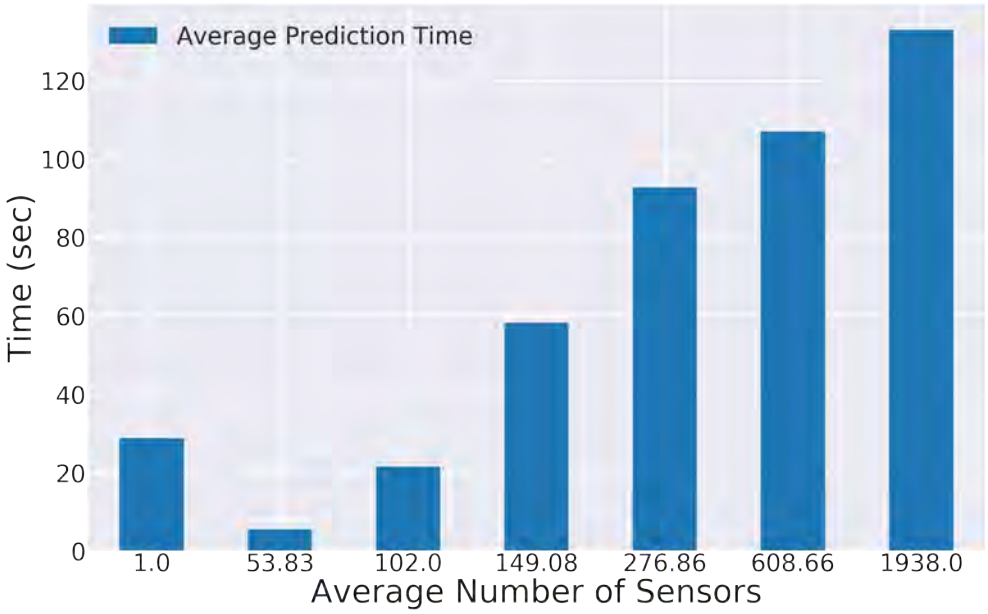


Figure 6.11: Relationship between prediction time and average number of traffic sensors

The rest of this section is divided into the following five subsections. The first subsection discusses the results from hyperparameter tuning the neural networks. The second subsection compares the measured and predicted traffic densities from the models. The third subsection evaluates the execution time for training and prediction of each neural network. The forth subsection illustrates difference in Root Mean Square Error between the models. The last subsection discusses the difference in accuracy when grouping traffic sensors by their location in a partition.

6.3.1 Hyperparameter Tuning

The figures 6.12 - 6.17 illustrate the results from performing hyperparameter tuning on the proposed traffic density prediction neural networks to determine the optimal number of hidden units used in the LSTM layers. The lines show the validation loss calculated as mean squared error (MSE) for each number of LSTM hidden units when training the traffic prediction models. The lower the MSE value, the better is the prediction accuracy of the model.

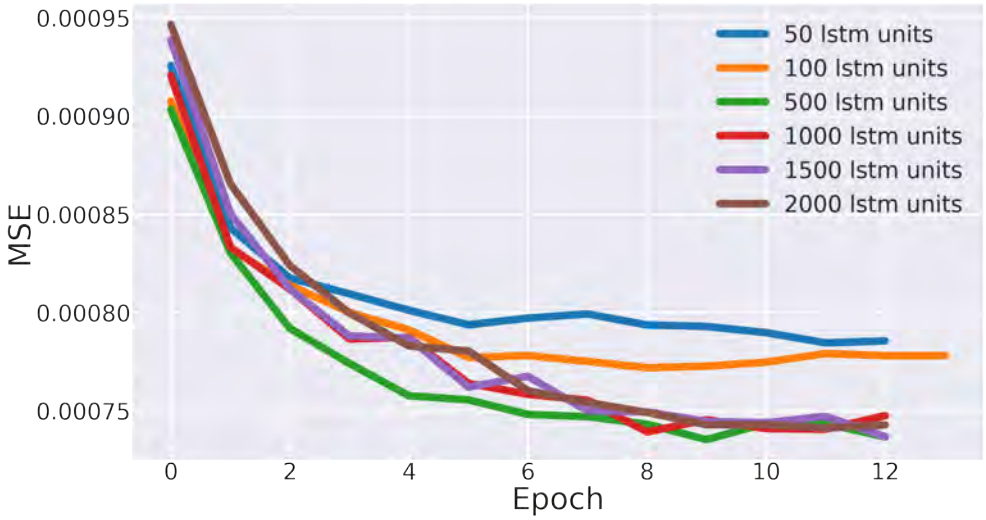


Figure 6.12: Validation loss for transportation system model

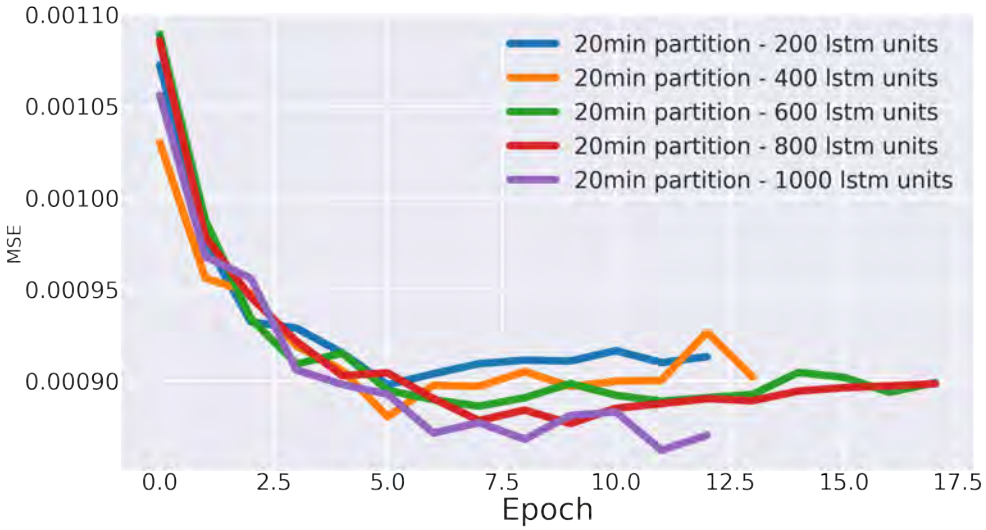


Figure 6.13: Validation loss for a 20 Minute partition model

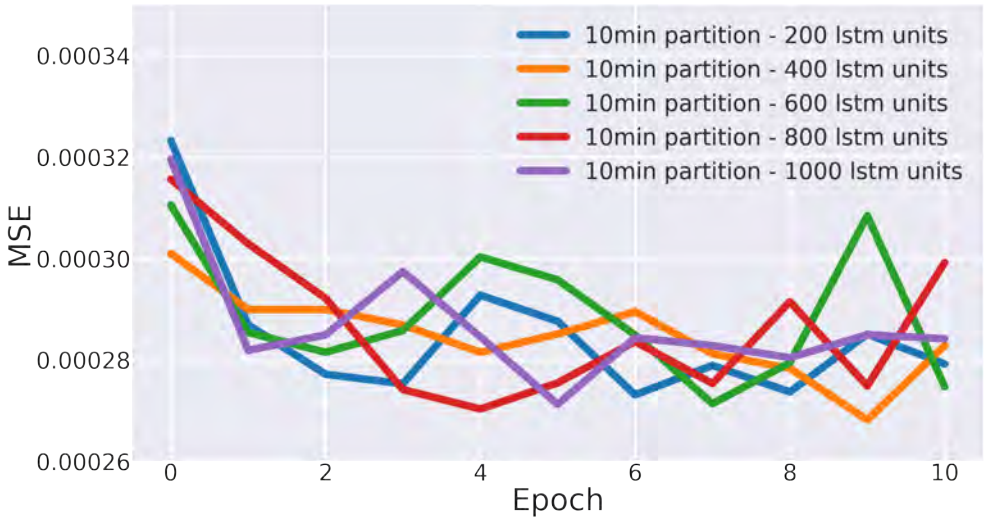


Figure 6.14: Validation loss for a 10 Minute partition model

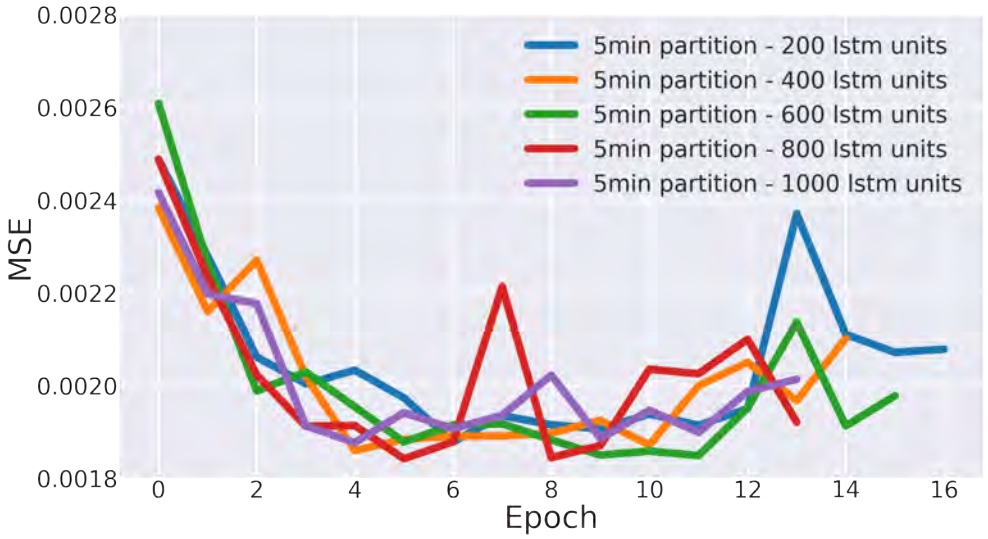


Figure 6.15: Validation loss for a 5 Minute partition model

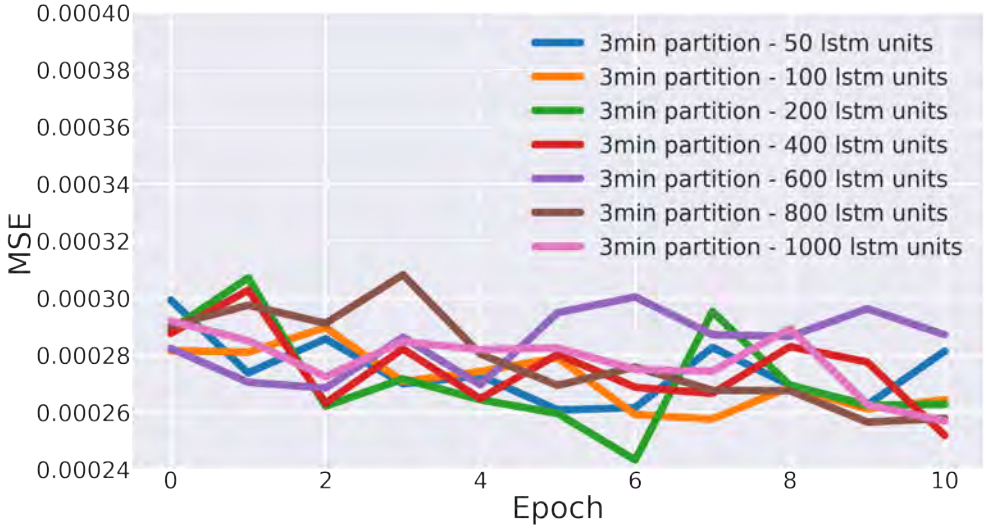


Figure 6.16: Validation loss for a 3 Minute partition model

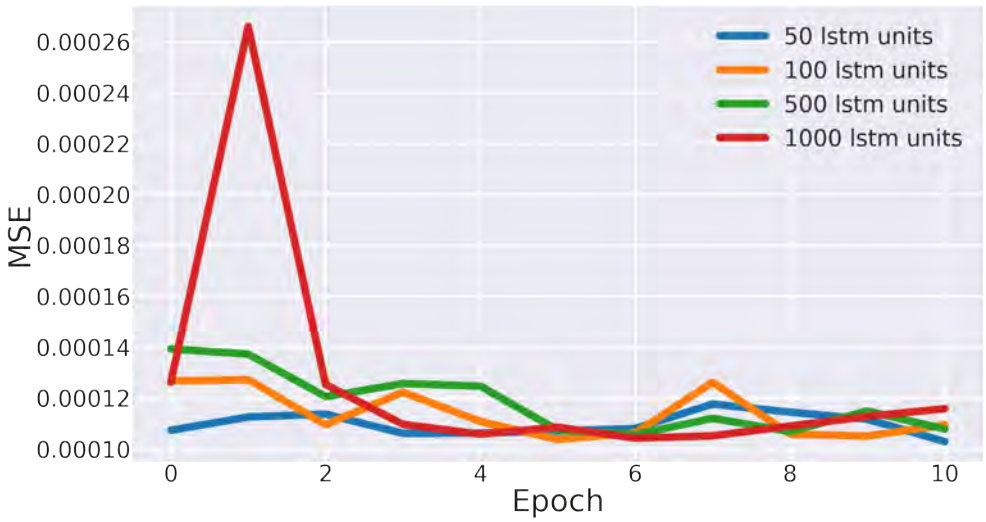


Figure 6.17: Validation loss for a single sensor model

Table 6.8 shows the number of LSTM hidden units that give the best accuracy after hyperparameter tuning the neural networks.

Neural Network	LSTM units
Whole Transportation System	1000
20 Minute Partitions	1000
10 Minute Partitions	800
5 Minute Partitions	600
3 Minute Partition	200
Single Sensor	50

Table 6.8: The number of LSTM hidden units producing the most accurate traffic predictions

Training Time And Model Complexity Comparison

Figures 6.18 - 6.23 show how the training time for each of the models proposed in section 4.4 increases as more LSTM hidden units are added to the neural networks.

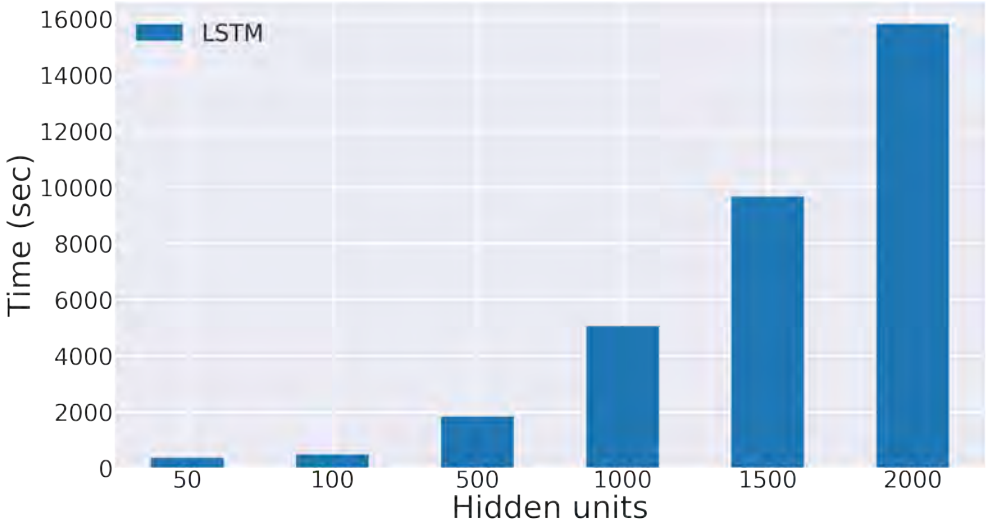


Figure 6.18: Relationship between model complexity and training time for the transportation system model

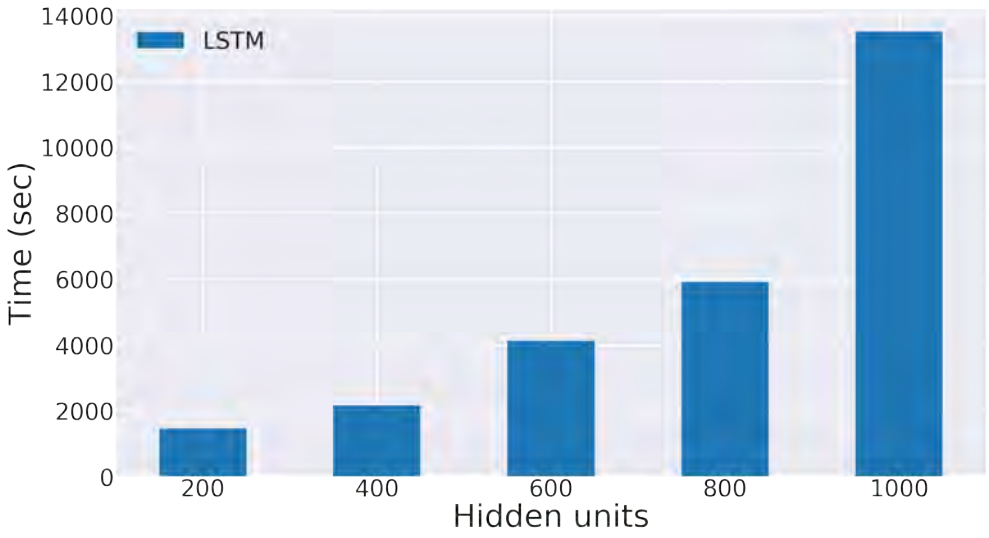


Figure 6.19: Relationship between model complexity and training time for the 20 minute partitioned transportation system model

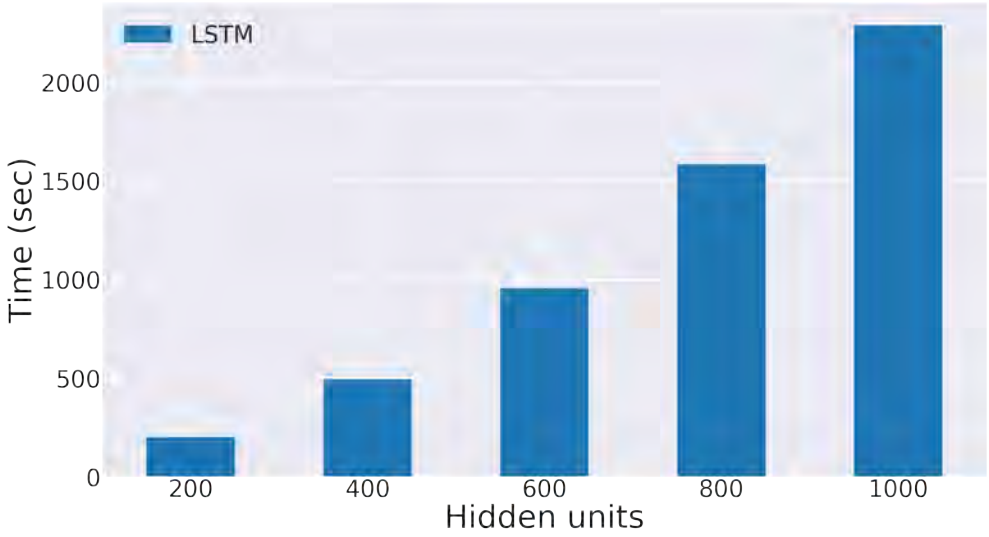


Figure 6.20: Relationship between model complexity and training time for the 10 minute partitioned transportation system model

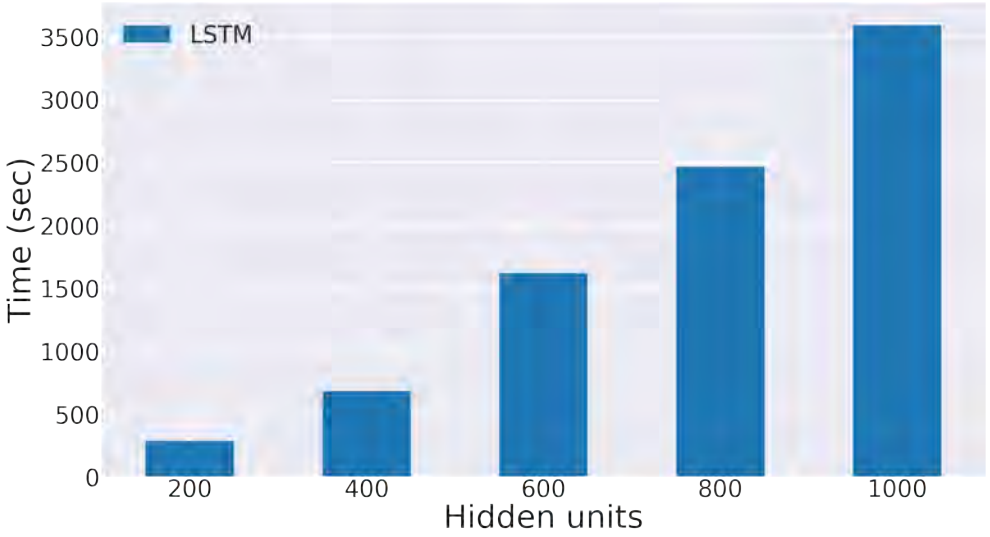


Figure 6.21: Relationship between model complexity and training time for the 5 minute partitioned transportation system model

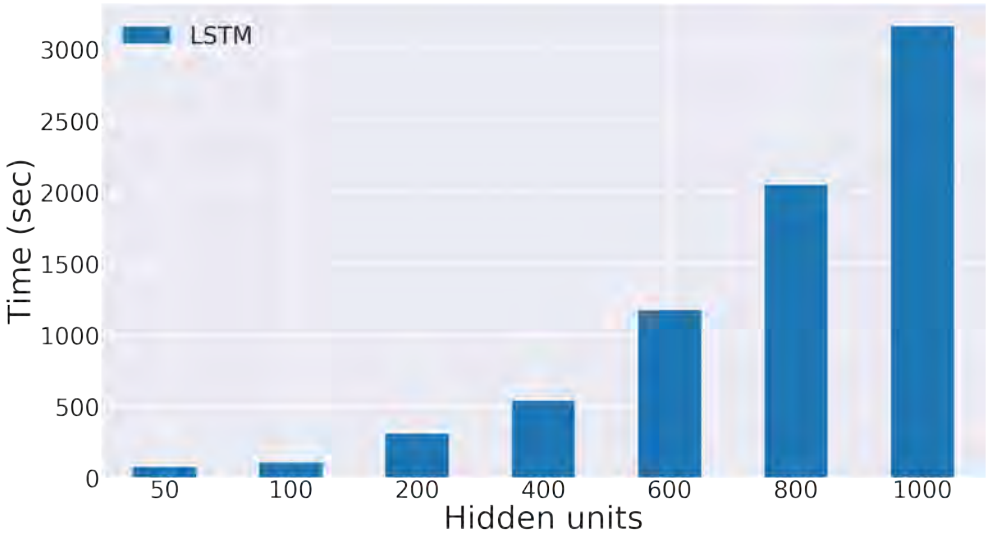


Figure 6.22: Relationship between model complexity and training time for the 3 minute partitioned transportation system model

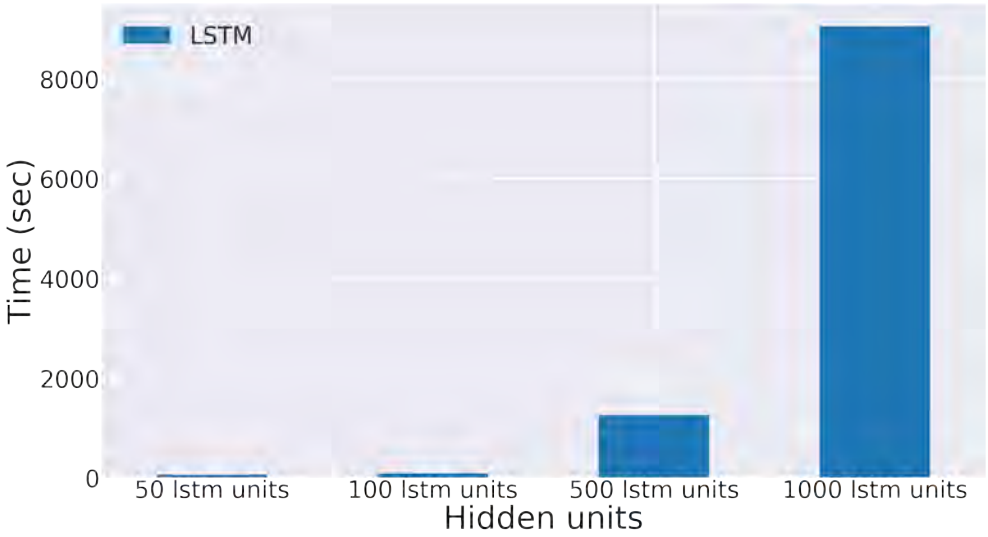


Figure 6.23: Relationship between model complexity and training time for the single sensor model

6.3.2 Measured and Predicted Data Evaluation

Table 6.9 shows the root mean squared error for 3, 5, 10, 20, 30 minute predictions for a random traffic sensor for each traffic density prediction model produced in section 5.3. Traffic sensor 1 at sensors' site *E265O 4,215* in Trafikverket's transportation system was selected.

Models	**t+3**	**t+5**	**t+10**	**t+20**	**t+30**
Single Sensor Models	1.35775	1.33077	1.36967	1.50176	1.65267
3 Minute Partition Models	1.33251	1.33428	1.3727	1.49484	1.58875
5 Minute Partition Models	1.29687	1.3158	1.32532	1.37804	1.4567
10 Minute Partition Models	1.30856	1.30035	1.31148	1.33656	1.38965
20 Minute Partition Models	1.35192	1.36733	1.3189	1.37145	1.37283
Transportation System Model	1.36274	1.3204	1.35993	1.37812	1.4066
Overlapping Partition Models	1.28561	1.31683	1.32171	1.37494	1.46496

Table 6.9: Root mean squared error difference between models for traffic sensor E265O-4215-1

Figures 6.24 - 6.30 draw the measured and predicted density values 30 minutes into the future for each of the traffic prediction models. As can be seen in the figures, the predicted value follows the measured

value indicating that all of the models are able to learn to varying degree of accuracy the temporal dependencies for the traffic sensor.

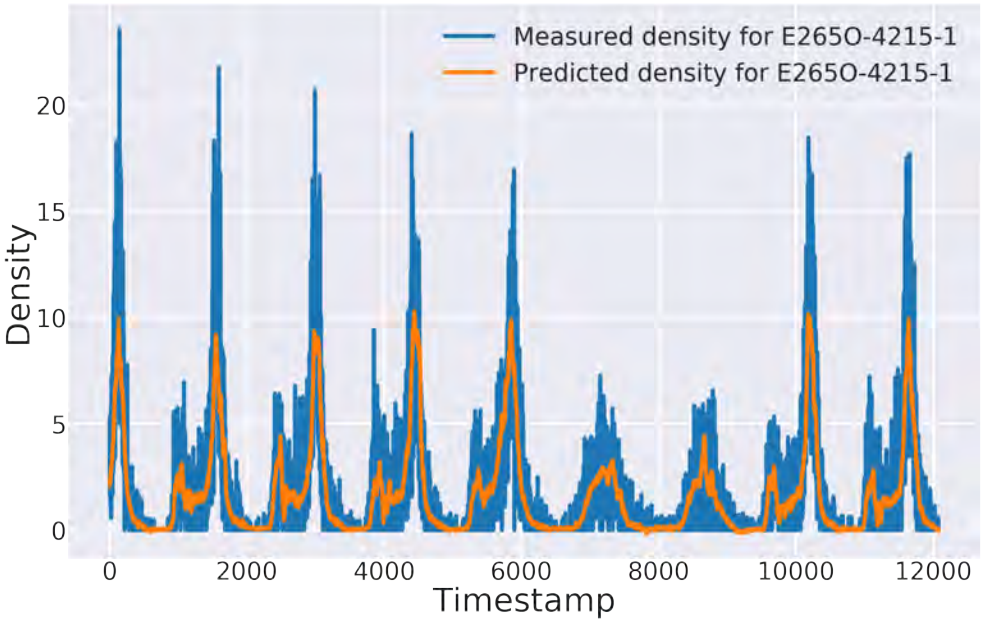


Figure 6.24: Comparison of measured and predicted density for the Transportation System Model

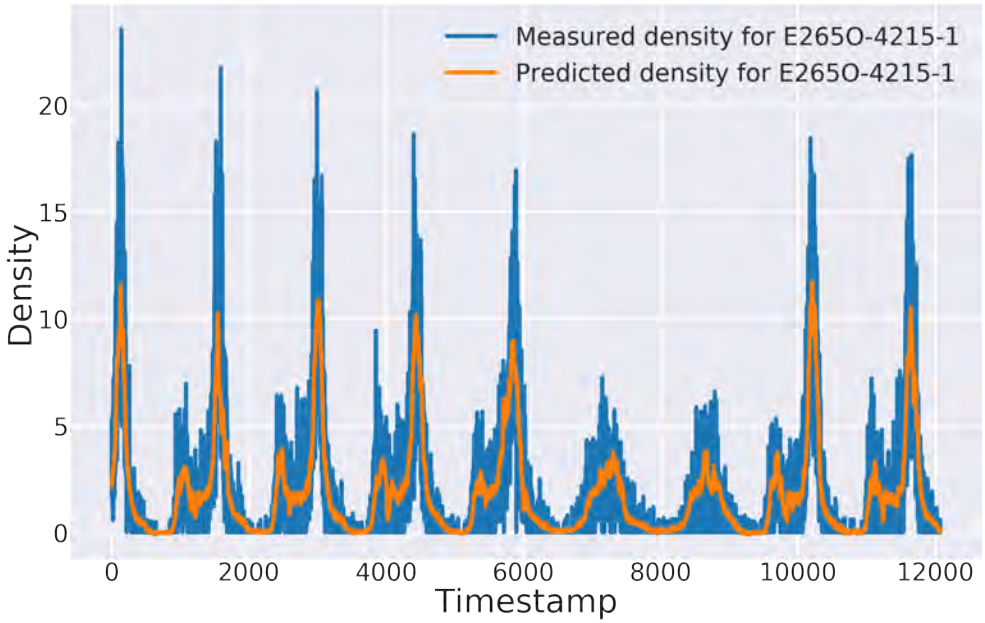


Figure 6.25: Comparison of measured and predicted density for the 20 Minute Partitioned Transportation System Models

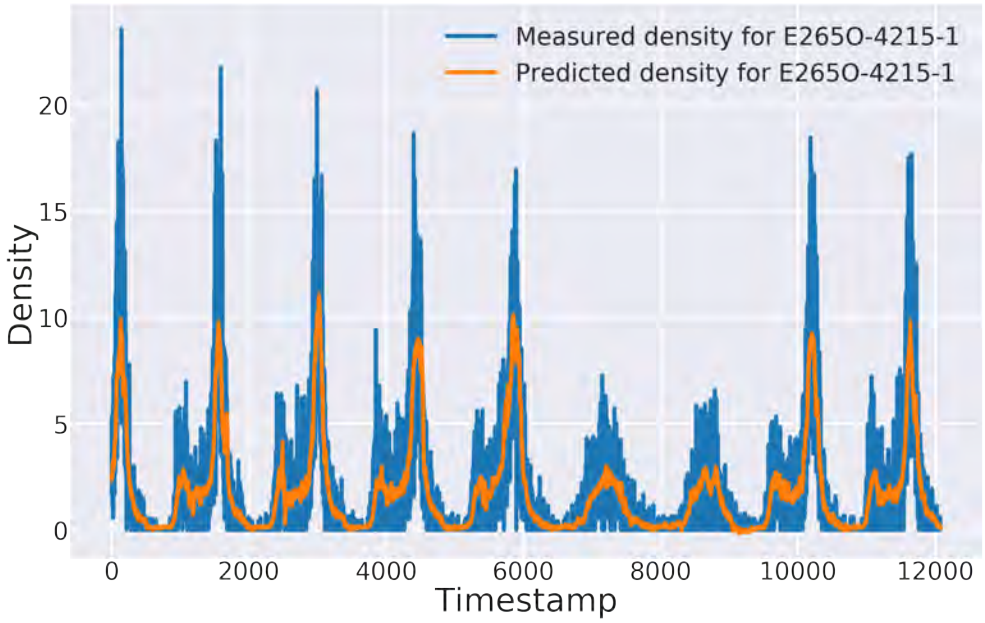


Figure 6.26: Comparison of measured and predicted density for the 10 Minute Partitioned Transportation System Models

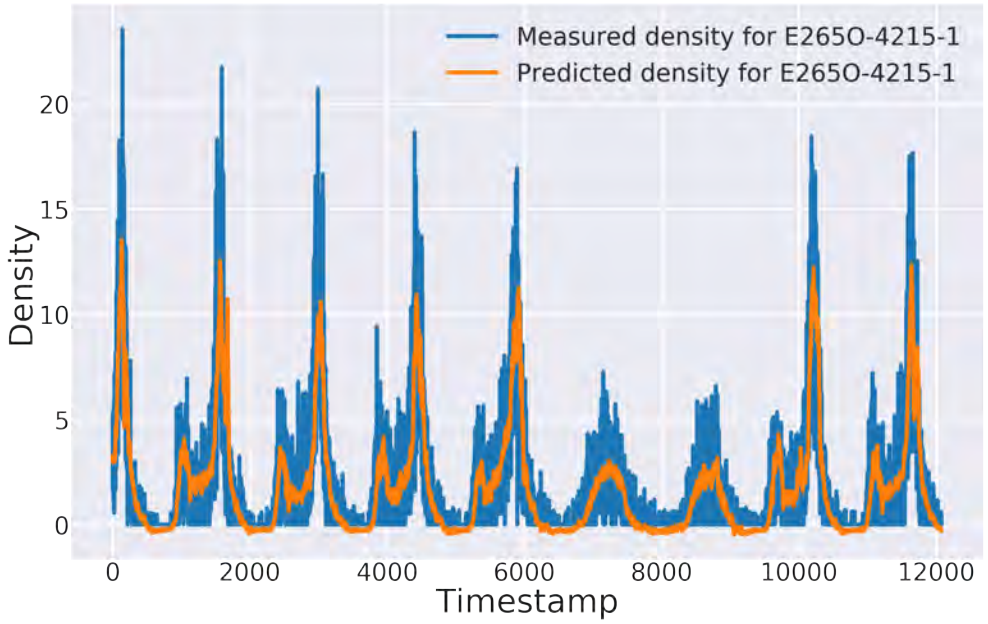


Figure 6.27: Comparison of measured and predicted density for the 5 Minute Partitioned Transportation System Models

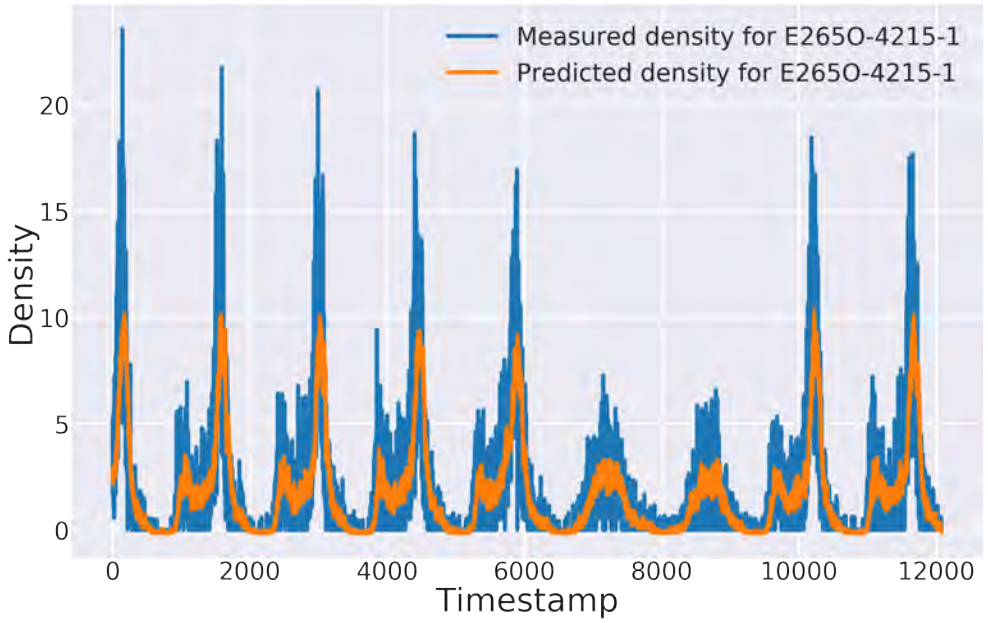


Figure 6.28: Comparison of measured and predicted density for the 3 Minute Partitioned Transportation System Models

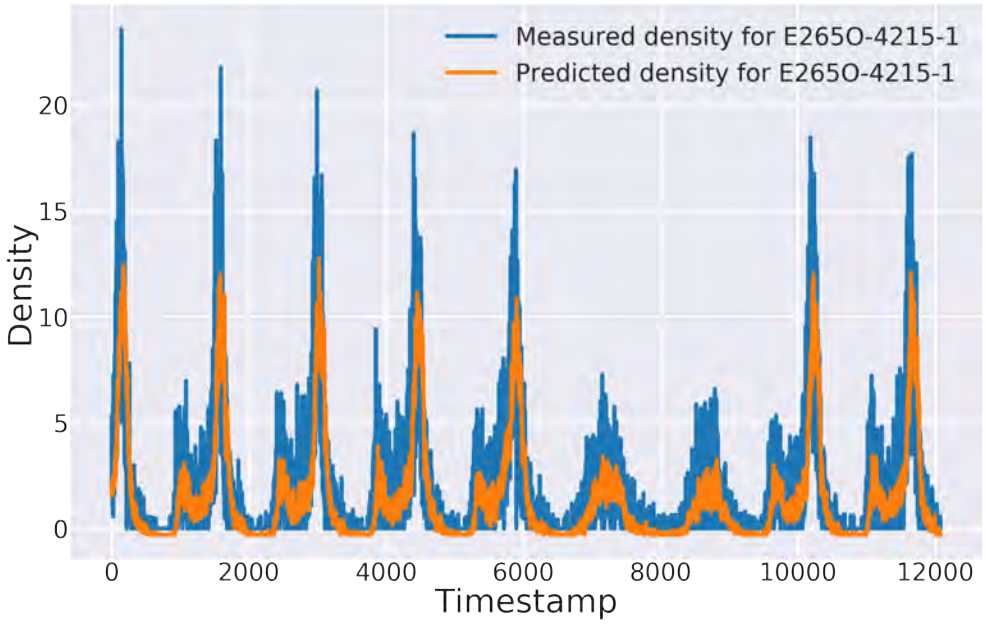


Figure 6.29: Comparison of measured and predicted density for Single Sensor Models

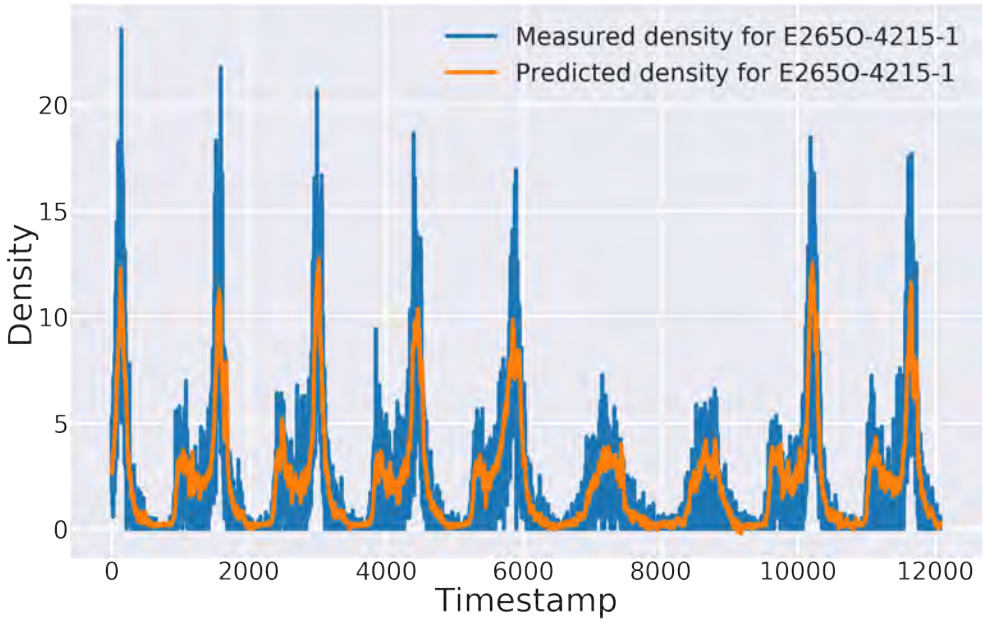


Figure 6.30: Comparison of measured and predicted density for Overlapping Partition Models

6.3.3 Time Evaluation

Important aspect of producing traffic prediction models is the amount of time it takes to train them and to make predictions. The resulting training and prediction times are dependent on the usage of the computing cluster as it is a shared resource and may be overloaded.

Figure 6.31 shows the average training time of each of the traffic density prediction models. The training time grows as the model complexity and input data increases. The Transportation System Model predicts for the largest number of traffic sensors and needs therefore the most LSTM hidden units and input data resulting in the longest training time. The Single Sensor Models need the shortest time for training as there is less information to be learned and the model contains the least number of LSTM hidden units and input data.

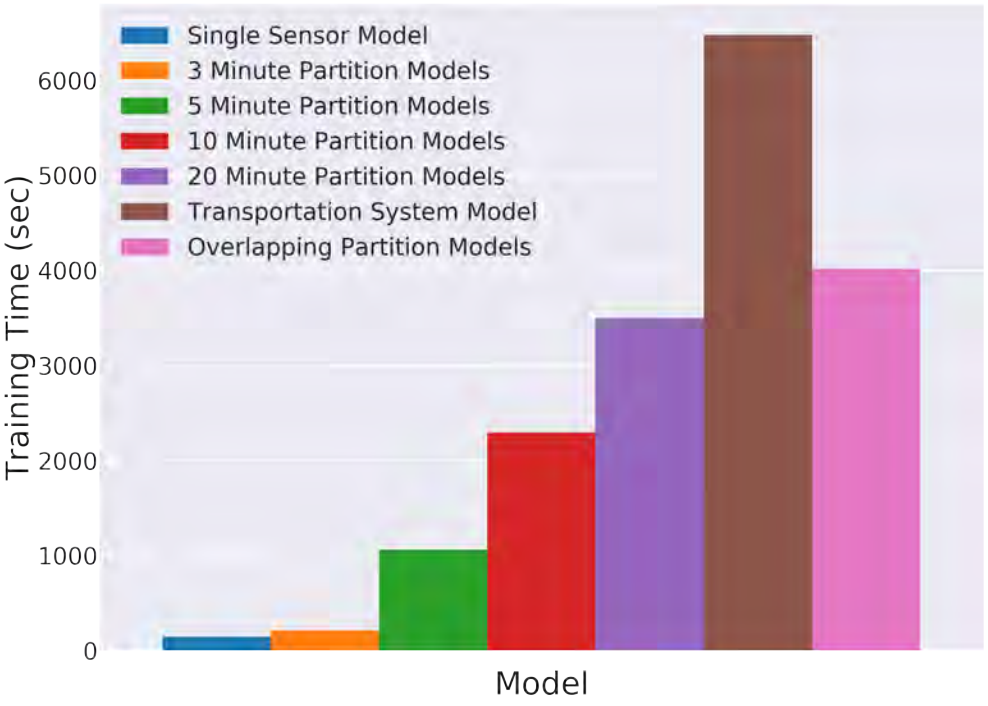


Figure 6.31: Average training time for each model

Figure 6.32 shows the total training time for all of the models when trained sequentially. Here the Single Sensor model towers over the others as each traffic sensor needs to be trained individually. The Overlapping Partition Models require as well a long training time as they are fed with large amount of information from surrounding traffic sen-

sors to predict for few traffic sensors. However, the training time can be reduced by training the models in parallel.

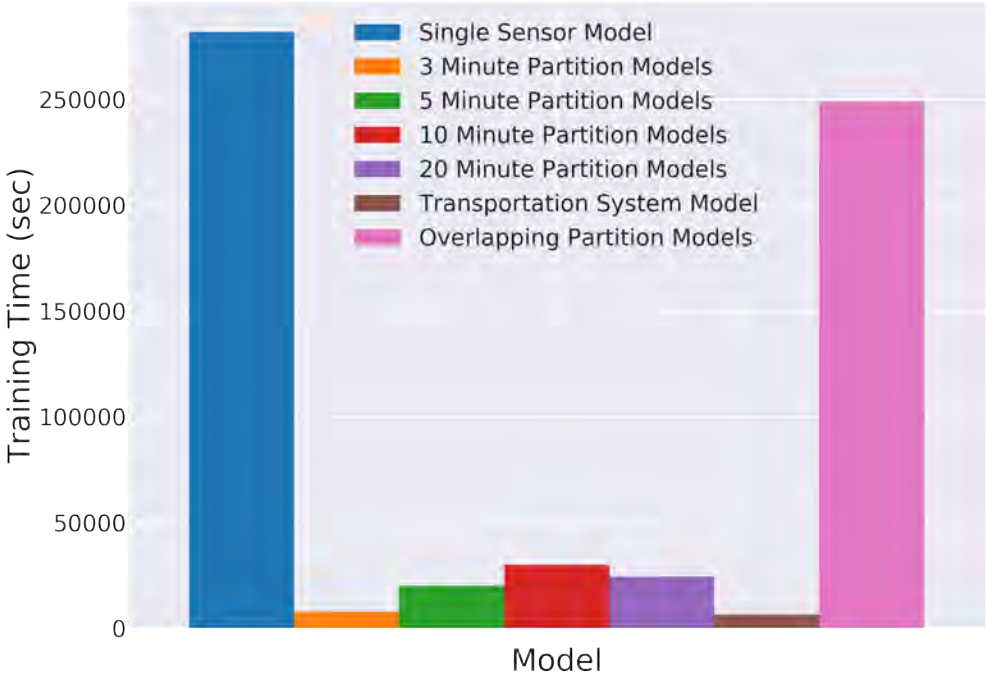


Figure 6.32: Total sequential training time for each model

Figure 6.33 demonstrates the average prediction time for each of the trained prediction models. The time varies from few seconds up to two minutes. Similarly to the training time, the prediction time for the transportation system model is the longest compared to the other models. The prediction time decreases as models use less input data and become less complex.

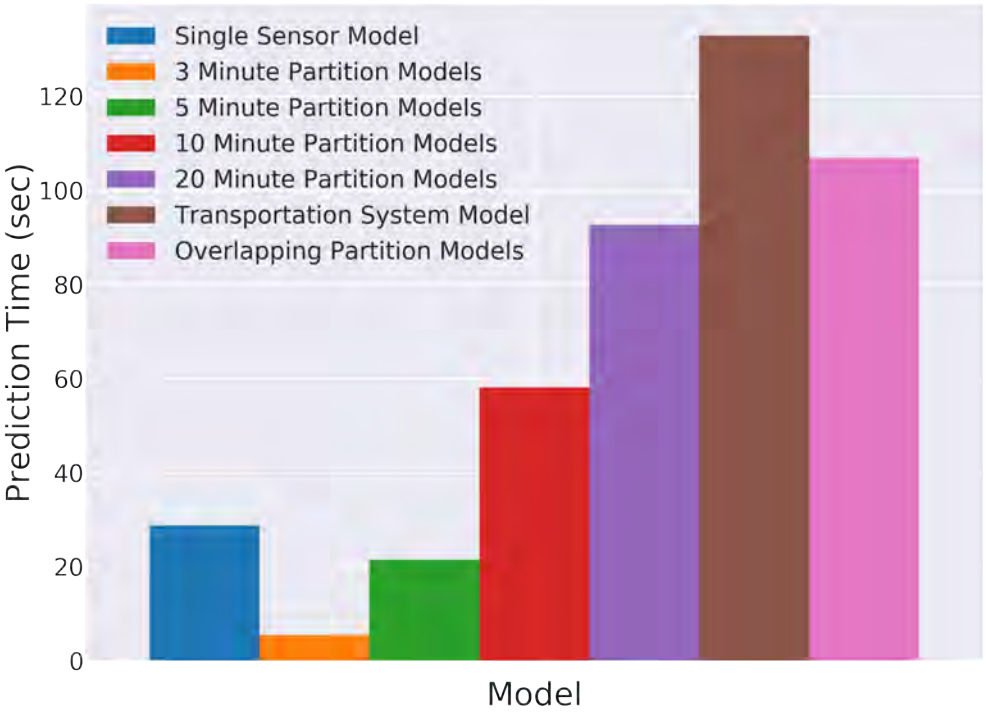


Figure 6.33: Average prediction time for each model

6.3.4 Root Mean Squared Error Evaluation

This subsection explores the average root mean squared error for each of the traffic density prediction models using the unseen test dataset. Calculating the root mean squared error indicates the accuracy of a prediction model. The lower the value, the better the prediction.

Table 6.10 and figure 6.34 illustrate how the root mean squared error grows as the models predict further into the future. On average, the Single Sensor Models give better prediction accuracy for short term predictions but produce the worst long term predictions, i.e. t+3 and t+30. The Transportation System Model produces the highest error for predictions lower than 20 minutes. Overall, the Overlapping Partition Models give on average the best results for all of the prediction times. A combination of these types of models can be used to achieve the most accurate predictions for the different predictions times.

The difference in error produced by the traffic prediction models is low demonstrating that partitioning a transportation system is a viable solution to allow for a scalable approach to traffic flow predictions

in an Intelligent Transportation System. The partitioned models have a reduced complexity as less input data and LSTM hidden units are needed, which improves prediction time. Additionally, these models can be moved to the edge of the transportation system located closer to the traffic sensors. This further reduces the prediction time as traffic measurements being sent travel shorter distances to the models.

Model	**t+3**	**t+5**	**t+10**	**t+20**	**t+30**
Single Sensor Models	3.73994	3.91913	4.22744	4.67089	5.00547
3 Minute Partition Models	3.91749	4.05118	4.30377	4.7146	4.98999
5 Minute Partition Models	3.97269	4.09059	4.30476	4.66408	4.9237
10 Minute Partition Models	4.13599	4.22315	4.4076	4.71508	4.89307
20 Minute Partition Models	4.26791	4.33955	4.50258	4.75038	4.90934
Transportation System Model	4.3455	4.39809	4.70794	4.72714	4.86186
Overlapping Partition Models	3.85881	3.9904	4.22765	4.5811	4.80663

Table 6.10: Average RMSE of predictions for all traffic sensors

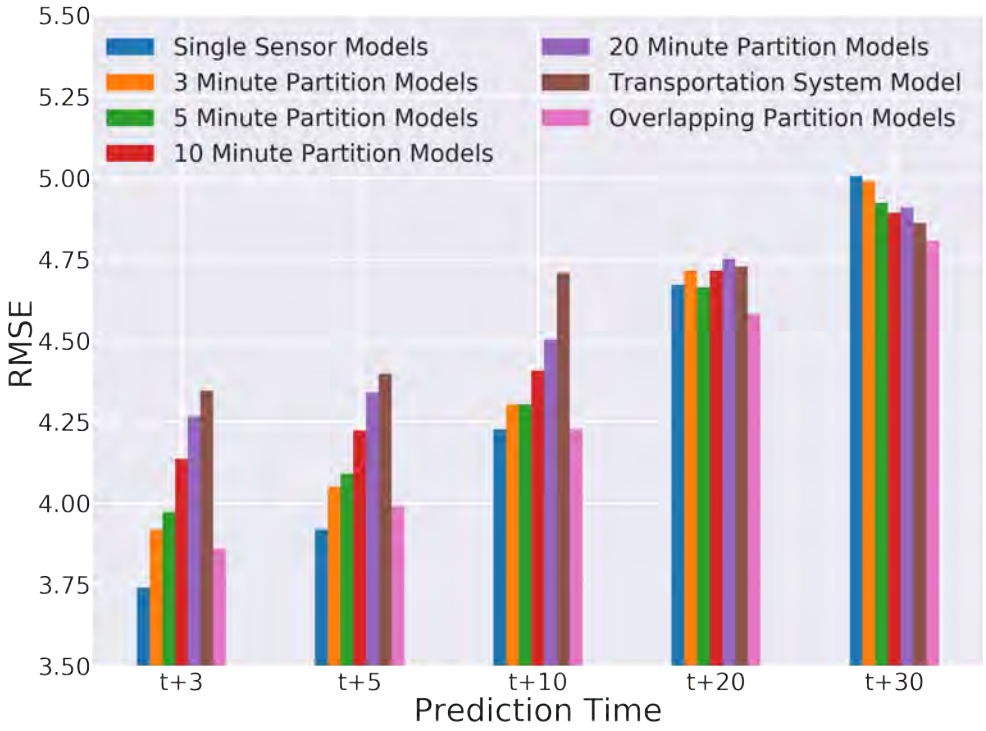


Figure 6.34: Comparison of average RMSE between all models for all traffic sensors

Figure 6.35 highlights further the benefits of using a partitioned transportation system for traffic predictions as a scalable solution. This can been seen from the box plots where the median is about the same as well as the interquartile range for all of the models.

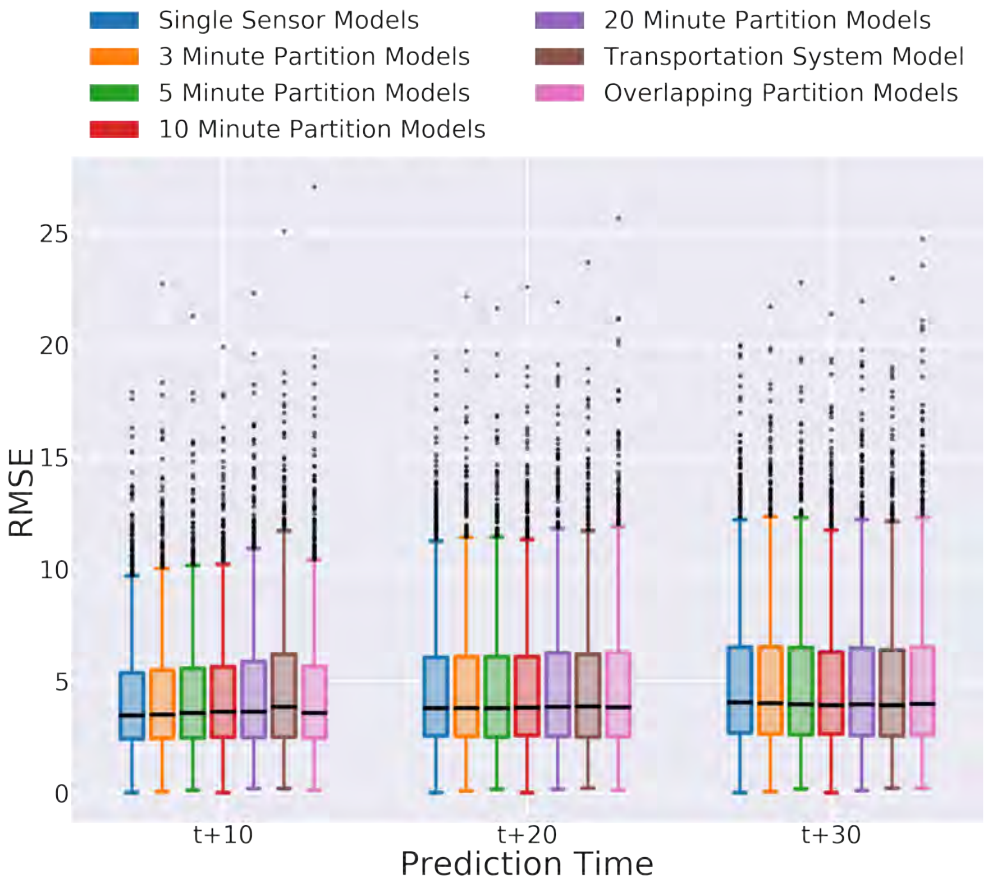


Figure 6.35: Comparison of average RMSE between all models for all traffic sensors

Figures 6.36 - 6.42 demonstrate the gradual increase in error as each of the models predict further into the future. The Single Sensor Models gives the best prediction accuracy for one minute predictions.

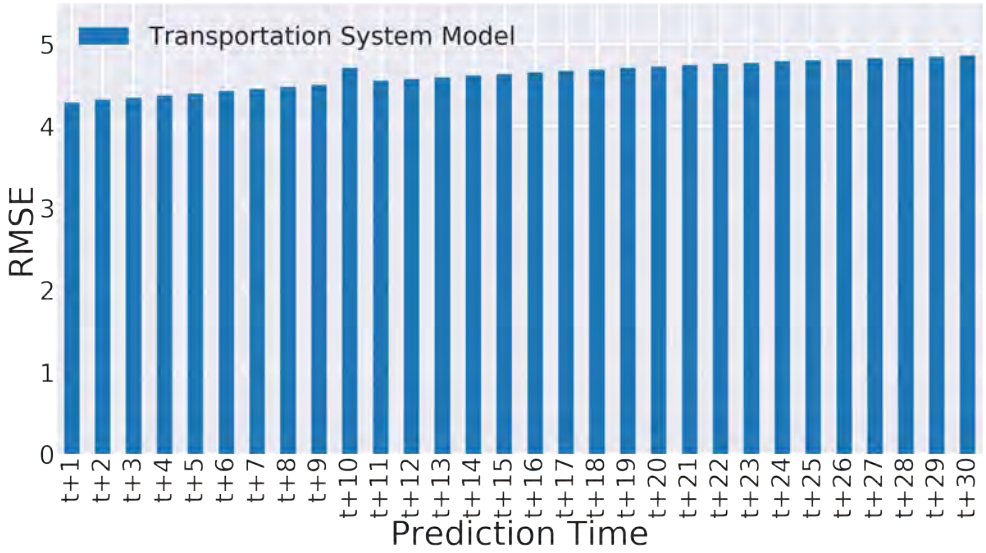


Figure 6.36: Average RMSE of all traffic sensors for the Transportation System Model

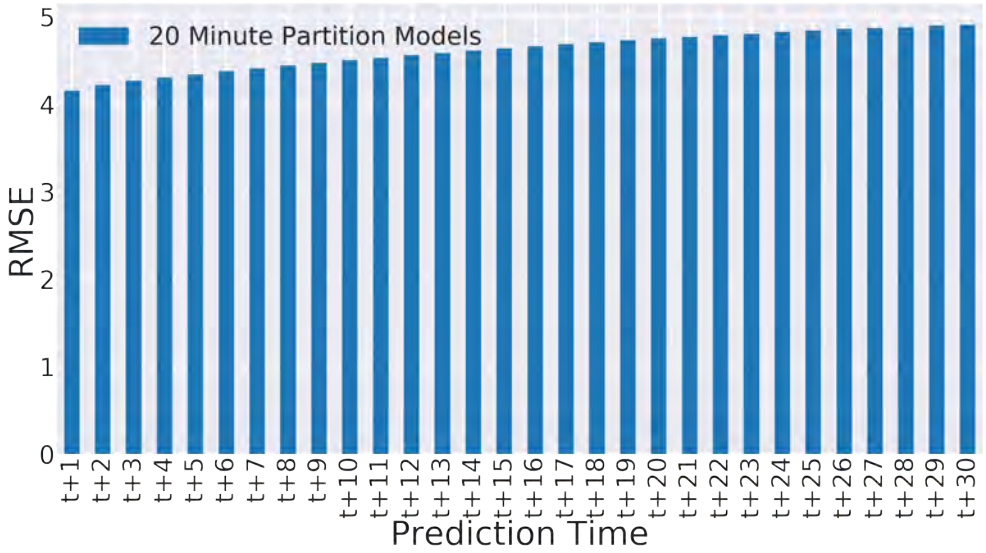


Figure 6.37: Average RMSE of all traffic sensors for the 20 Minute Partition System Models

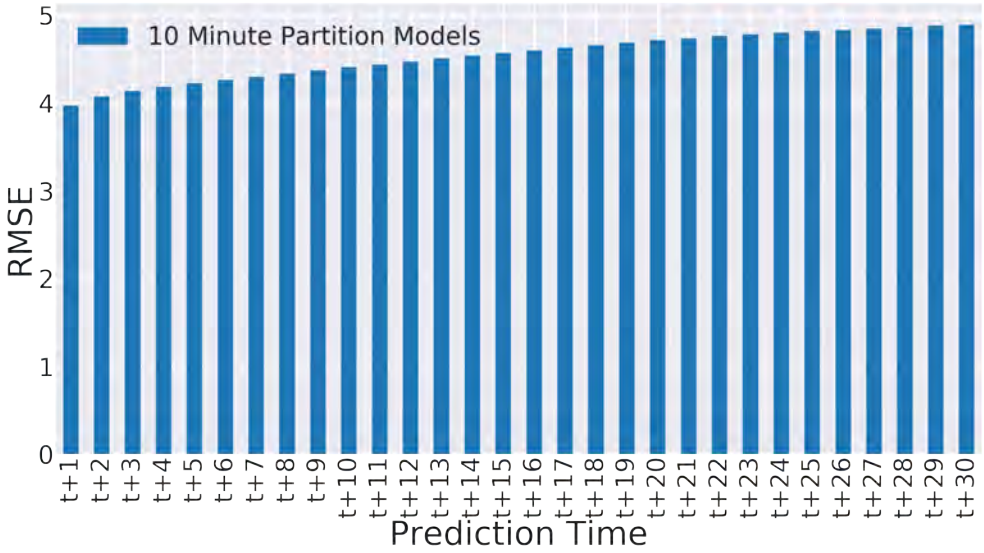


Figure 6.38: Average RMSE of all traffic sensors for the 10 Minute Partition System Models

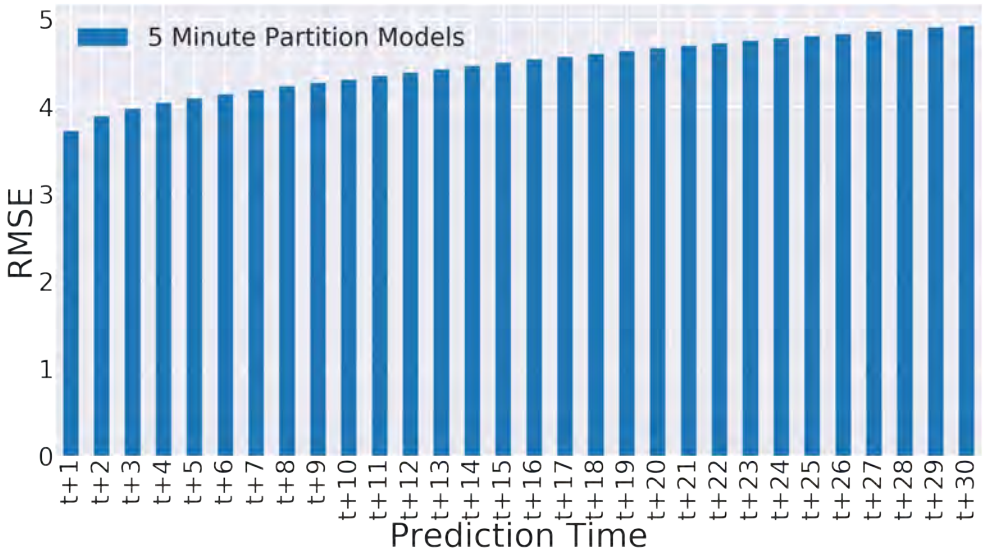


Figure 6.39: Average RMSE of all traffic sensors for the 5 Minute Partition System Models

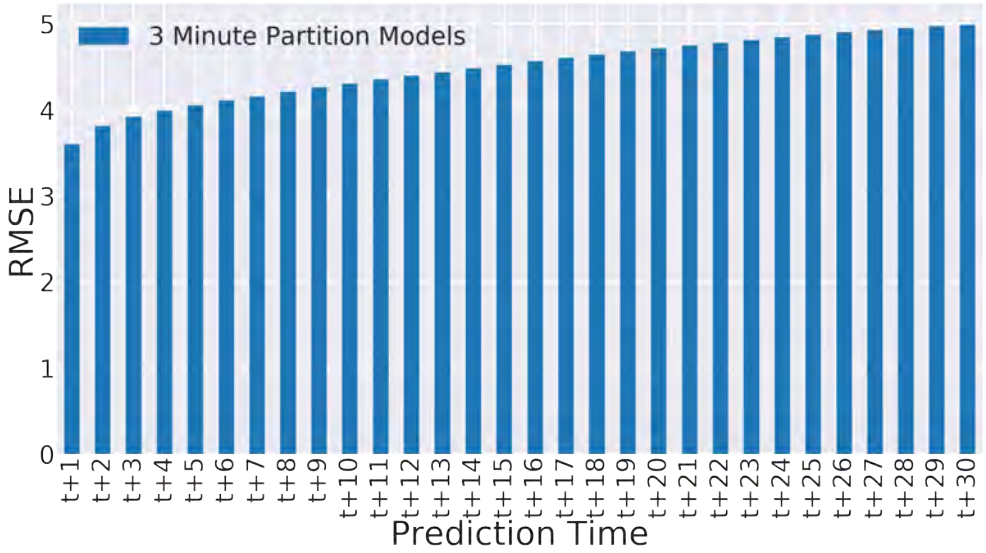


Figure 6.40: Average RMSE of all traffic sensors for the 3 Minute Partition System Models

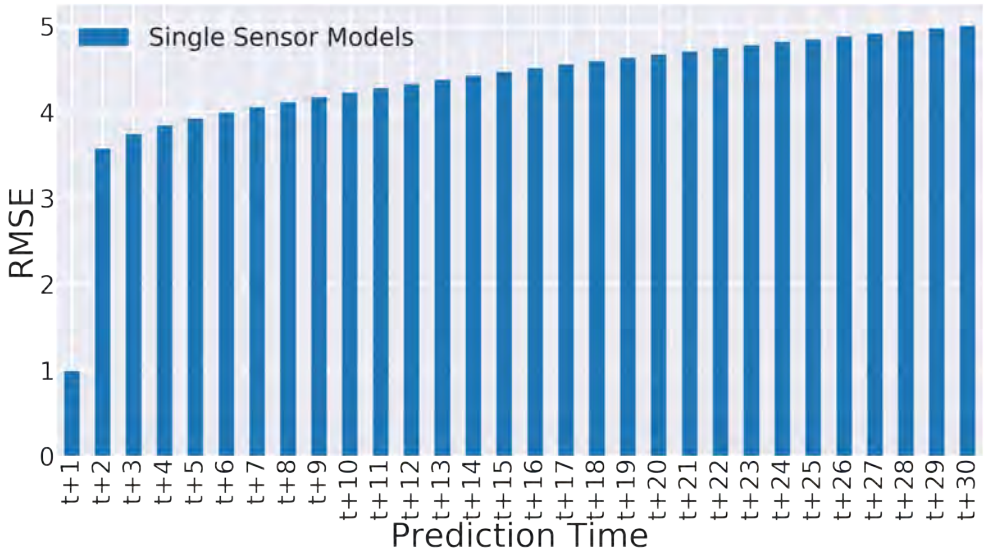


Figure 6.41: Average RMSE of all traffic sensors for the Single Sensor Models

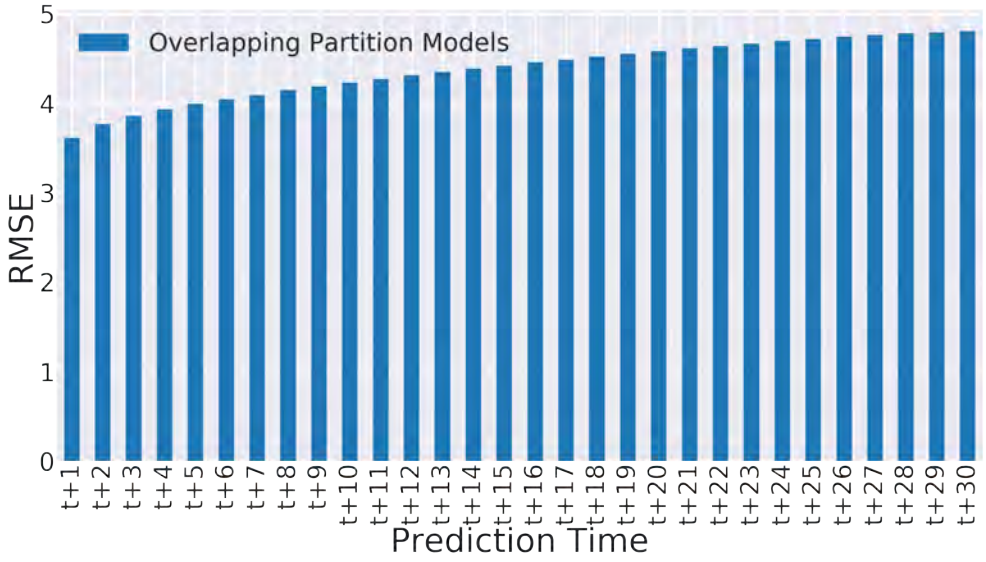


Figure 6.42: Average RMSE of all traffic sensors for the Overlapping Partition Models

6.3.5 Sensor Group Evaluation within a Partition

Traffic sensors at entrances of a partition should demonstrate higher prediction error compared to traffic sensors located in the middle and at the exits of a partition. This is because the models have more spatial dependency information for the traffic sensors located closer to the exits of the partition.

Table 6.11 shows in principal that the predictions for the traffic sensors located at the start of the partitions (start group) have a higher root mean squared error while the prediction error for the traffic sensors located at the end of the partition (end group) are the lowest. The root mean squared error of the predictions for the rest of the traffic sensors (center group) is between the other two groups. However, the root mean squared error for the center group is in some cases higher than for the start group. This may be due to the large size of the center group compared to the other groups. The center group encompasses the majority of all traffic sensors, except for the traffic sensors located directly at entrances and exits of a partition.

Model	Groups	t+10	t+20	t+30
3 Minute Partition Models	Start	3.473036	3.747368	4.013529
	Center	3.515262	3.796037	4.027587
	End	3.408933	3.475926	3.526219
5 Minute Partition Models	Start	3.721472	3.980659	4.261294
	Center	3.559528	3.780412	3.947681
	End	3.431928	3.582781	3.720938
10 Minute Partition Models	Start	3.711675	3.799795	3.932735
	Center	3.645098	3.828238	3.926750
	End	3.195844	3.359010	3.364163
20 Minute Partition Models	Start	3.392057	3.538455	3.660241
	Center	3.693661	3.891483	3.988295
	End	3.092131	3.207516	3.211633

Table 6.11: Median of the average traffic prediction RMSE for each traffic sensor group

Figures 6.43 - 6.46 illustrate the results of grouping traffic sensors into three groups. A start group consists of traffic sensors located at the entrances of a partition. A center group includes traffic sensors located in the middle of a partition. A end group consists of traffic sensors located at exits of a partition. The figures show the difference in average root mean squared error for each of the three groups with the prediction time 10, 20 and 30 minutes. The results demonstrate that the accuracy is highest for traffic sensors in the end group, while traffic sensors in the center and start group produce lower prediction accuracy. This becomes more noticeable as the prediction time increases from 10 to 30 minutes. The results suggest that the models are able to learn that the traffic conditions at entrances of the partition propagate to the exists of the partition.

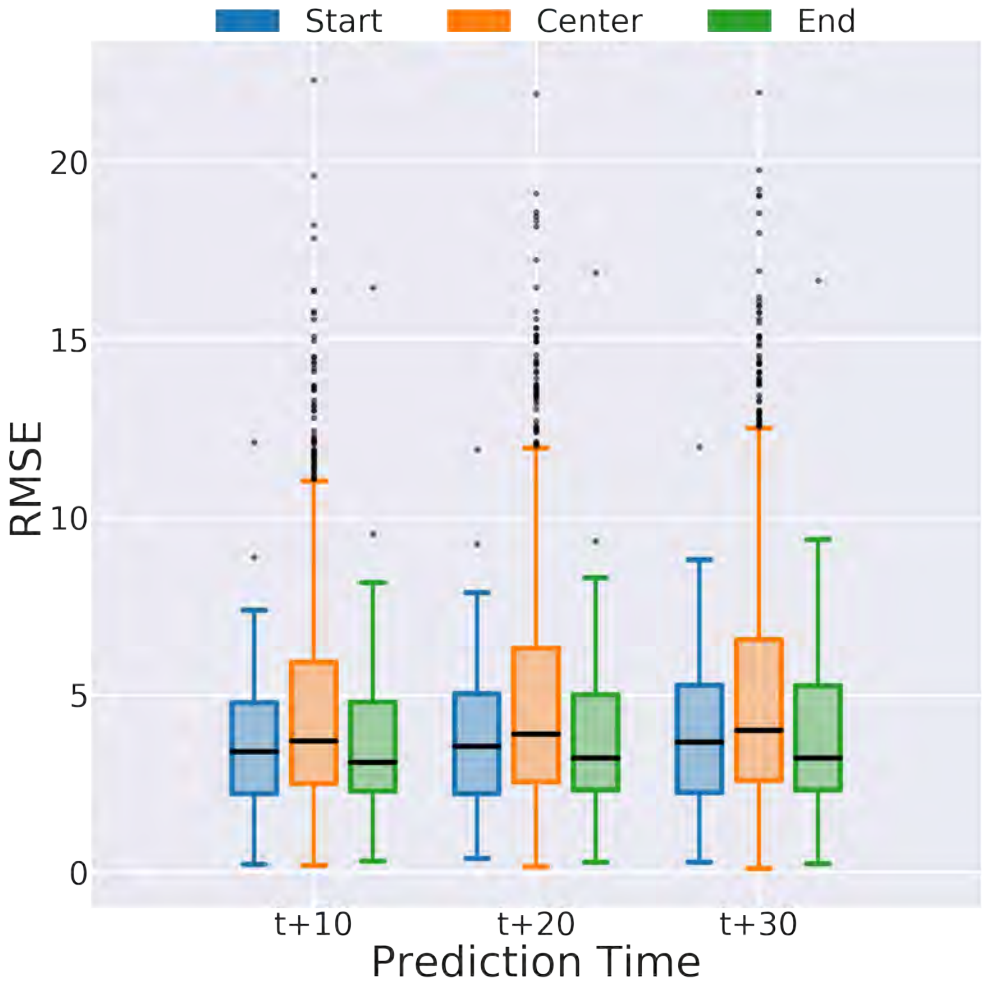


Figure 6.43: Average RMSE comparison between sensor groups in the 20 Minute Partition System Models

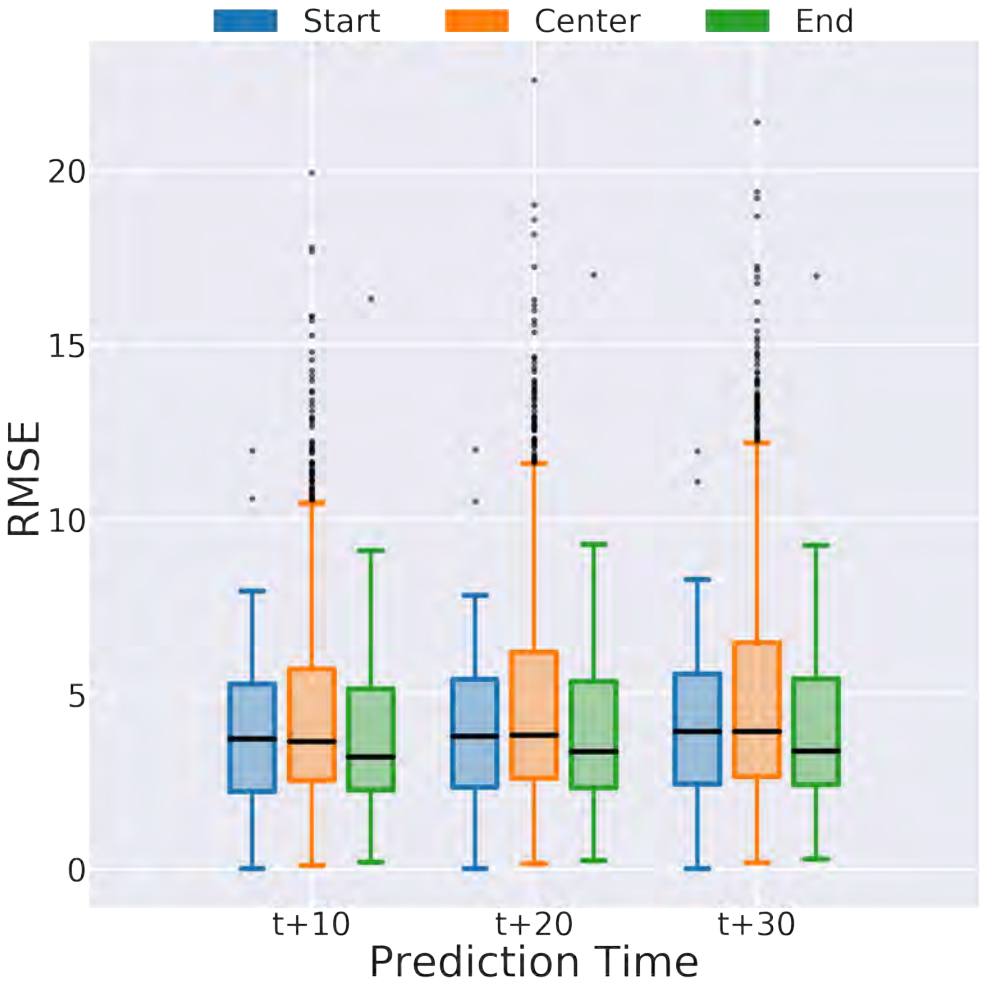


Figure 6.44: Average RMSE comparison between sensor groups in the 10 Minute Partition System Models

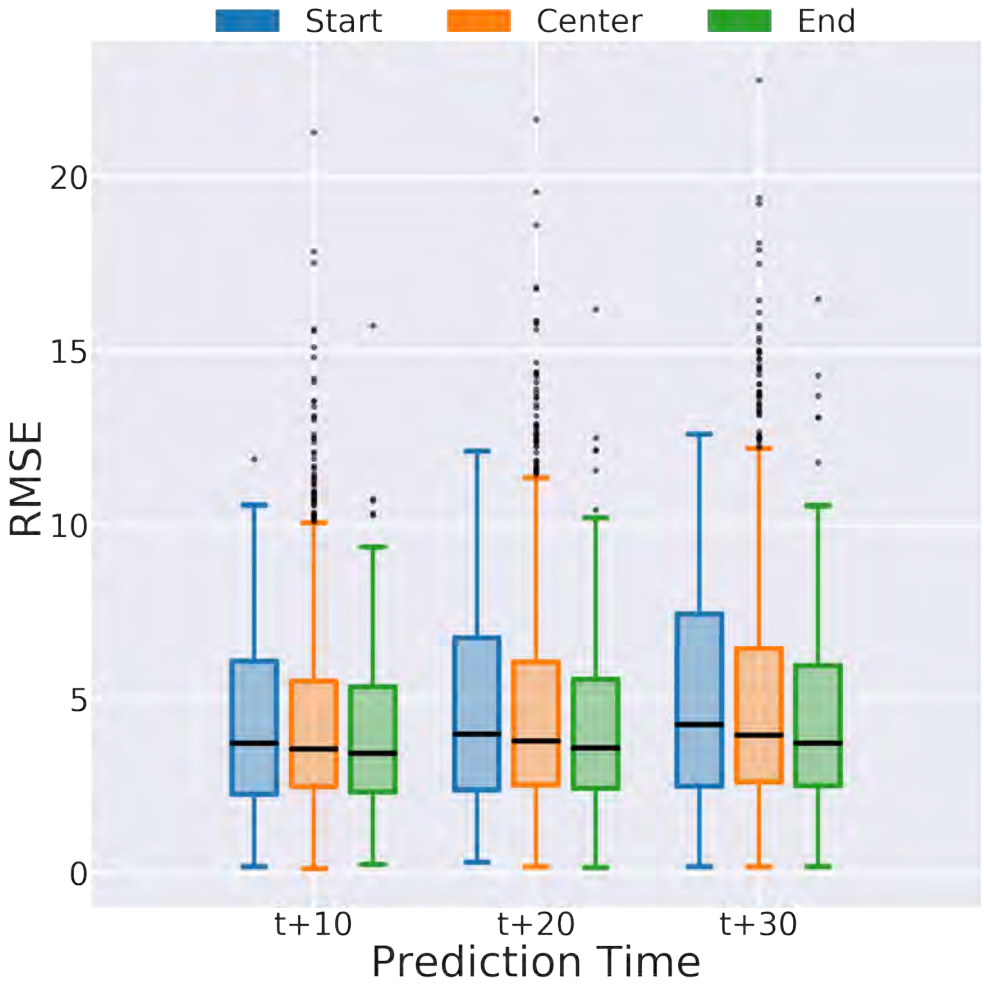


Figure 6.45: Average RMSE comparison between sensor groups in the 5 Minute Partition System Models

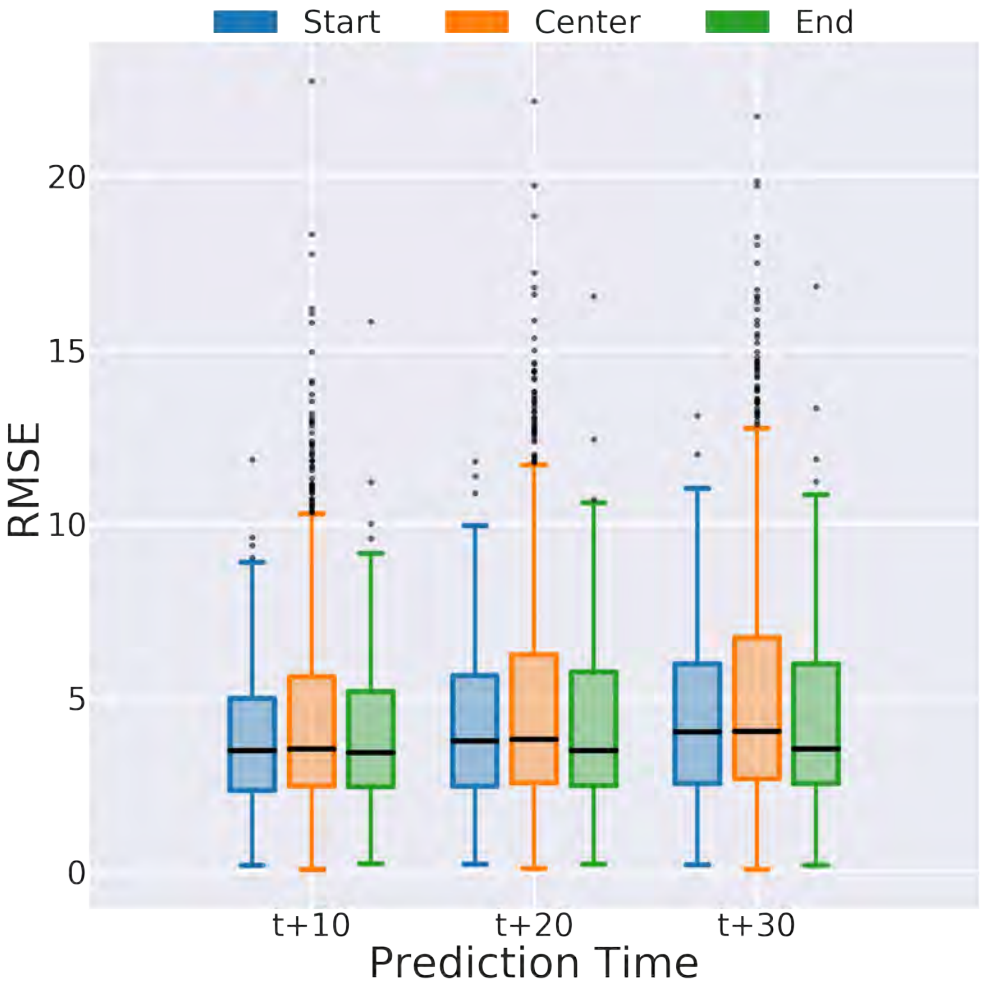


Figure 6.46: Average RMSE comparison between sensor groups in the 3 Minute Partition System Models

The figures in the rest of the subsection compare the root mean squared error produced by each of the traffic sensor groups in the 3, 5, 10 and 20 Minute Partition Models with the same traffic sensors in the Single Sensor Models, the Transportation System Model and the Overlapping Partition Models.

20 Minute Partition Group Comparison

The figures 6.47 - 6.49 indicate that the 20 minute partition models give the best average prediction accuracy compared to the other three models for all of traffic sensor groups, except for the start group where the single sensor model has the lowest root mean squared error. This may be due to the fact that the 20 Minute model does not have any traffic measurements from the traffic sensors located before the start group.

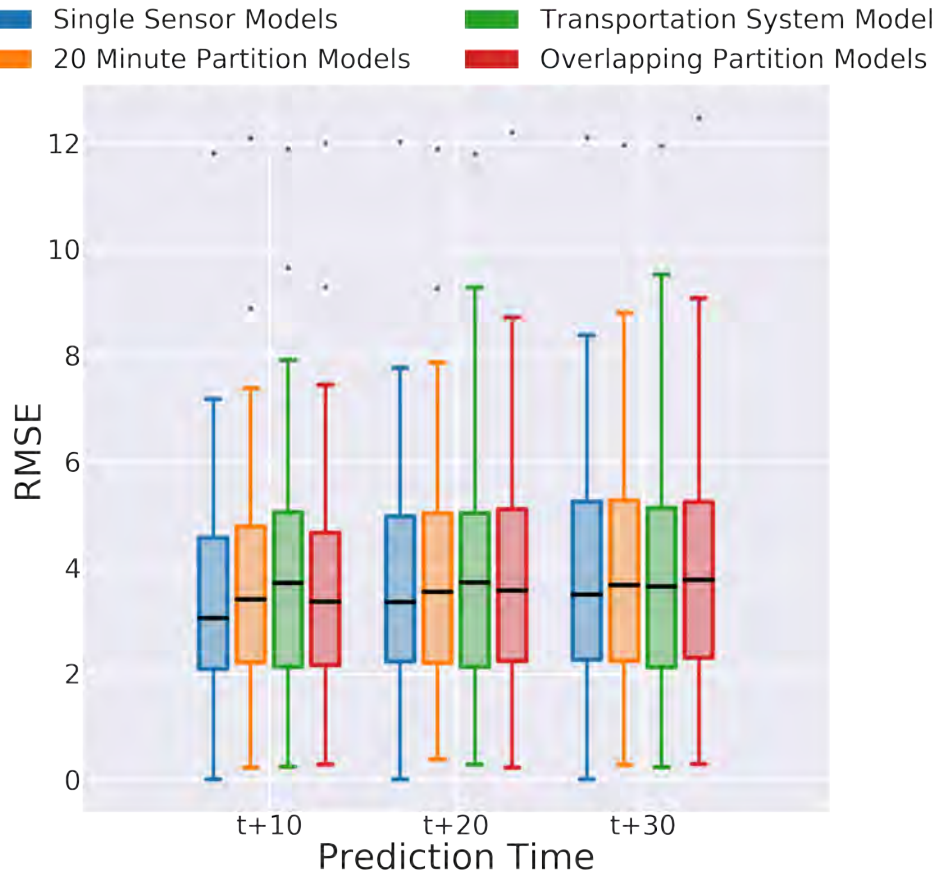


Figure 6.47: Comparison of average RMSE of traffic sensors in the start group in the 20 Minute Partition System Models

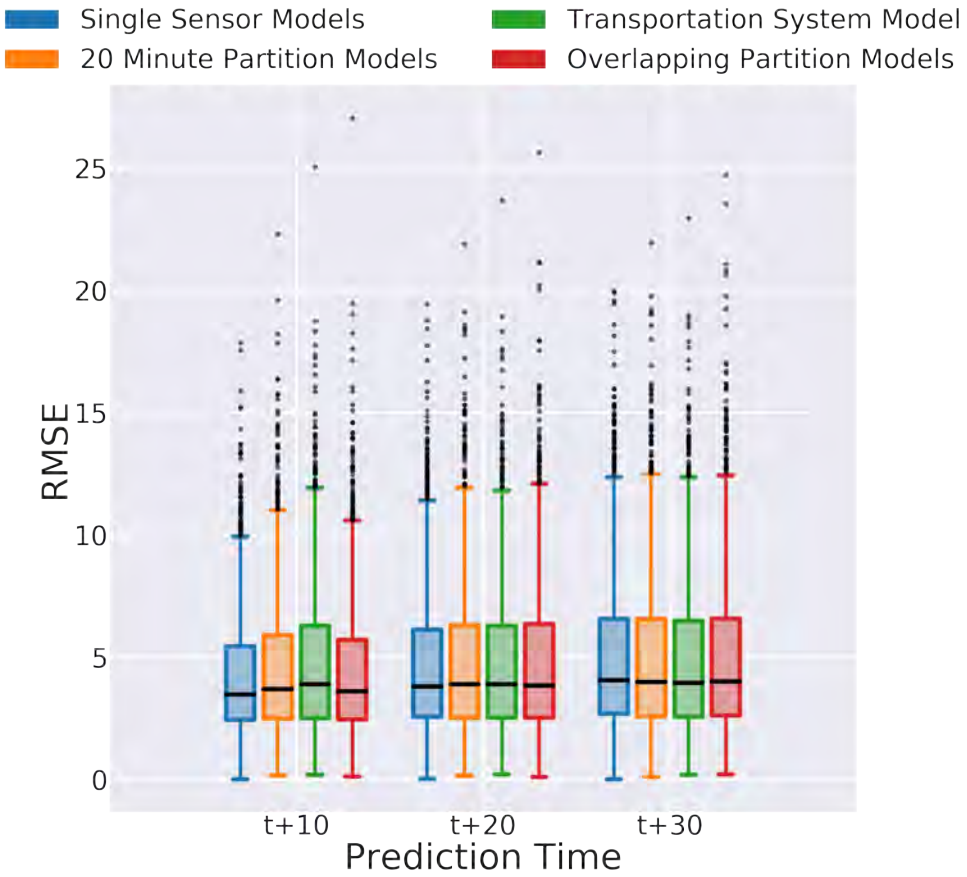


Figure 6.48: Comparison of average RMSE of traffic sensors in the center group in the 20 Minute Partition System Models

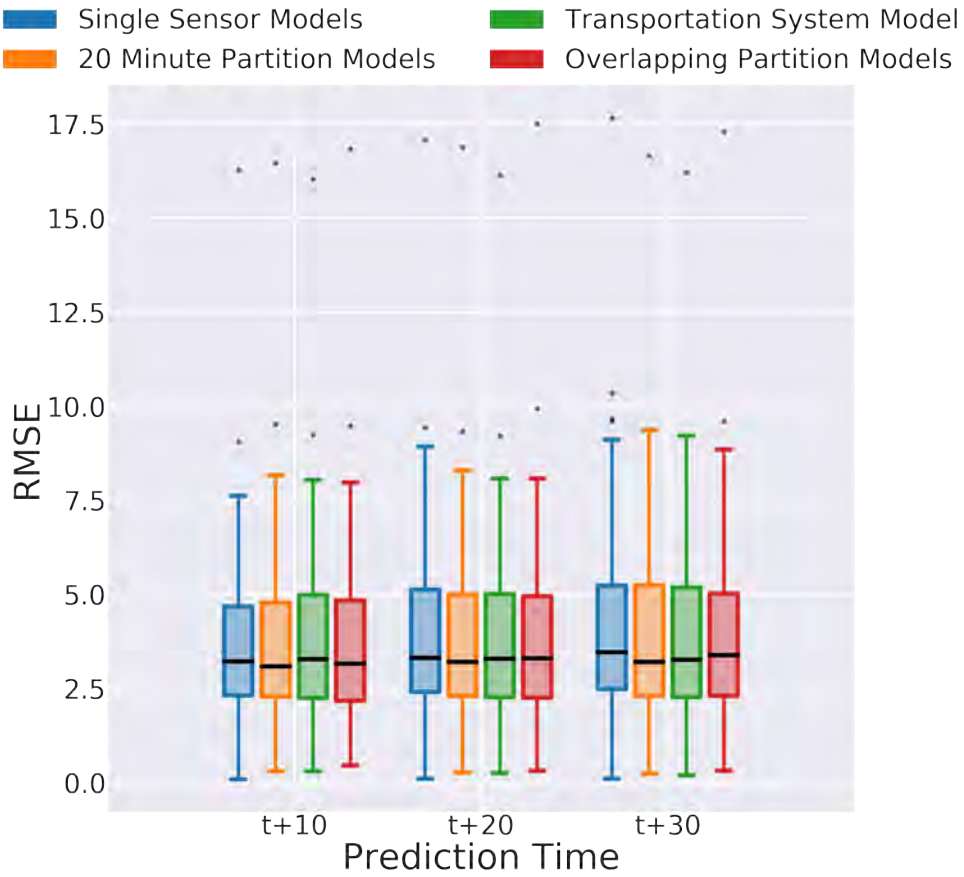


Figure 6.49: Comparison of average RMSE of traffic sensors in the end group in the 20 Minute Partition System Models

10 Minute Partition Group Comparison

The figures 6.50 - 6.52 indicate that the 10 minute partition models give the best average prediction accuracy compared to the other three models for all of traffic sensor groups, except for the start group where the single sensor model has the lowest root mean squared error.

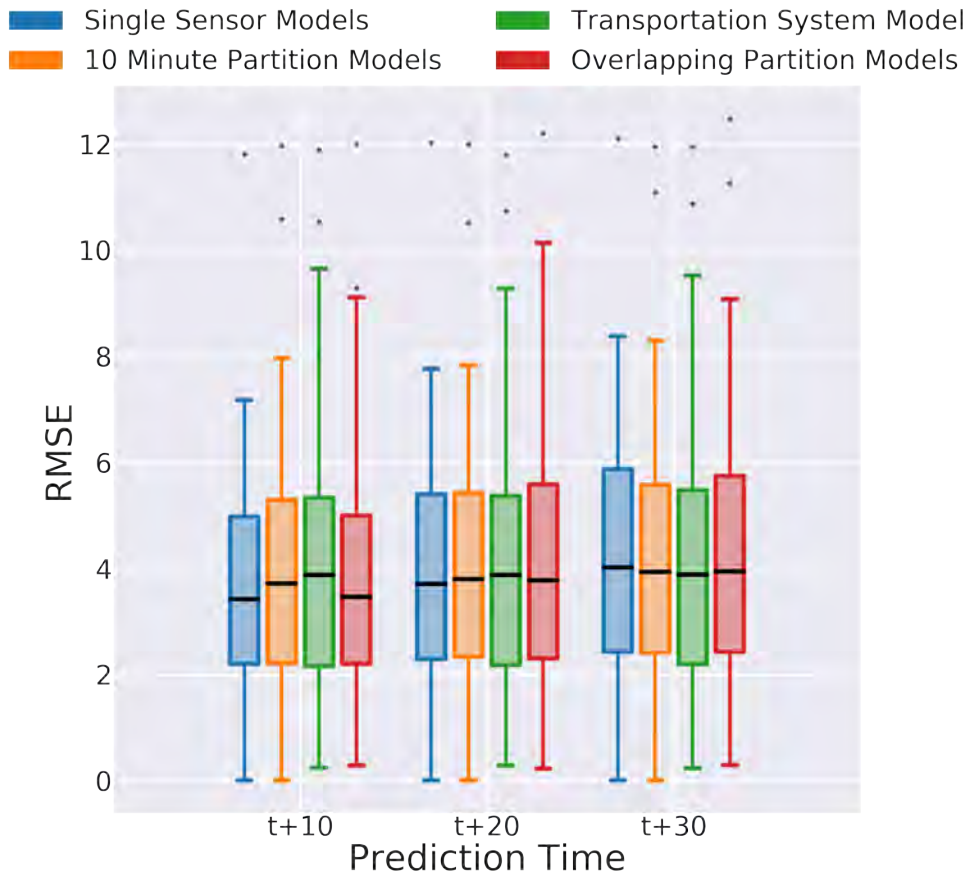


Figure 6.50: Comparison of average RMSE of traffic sensors in the start group in the 10 Minute Partition System Models

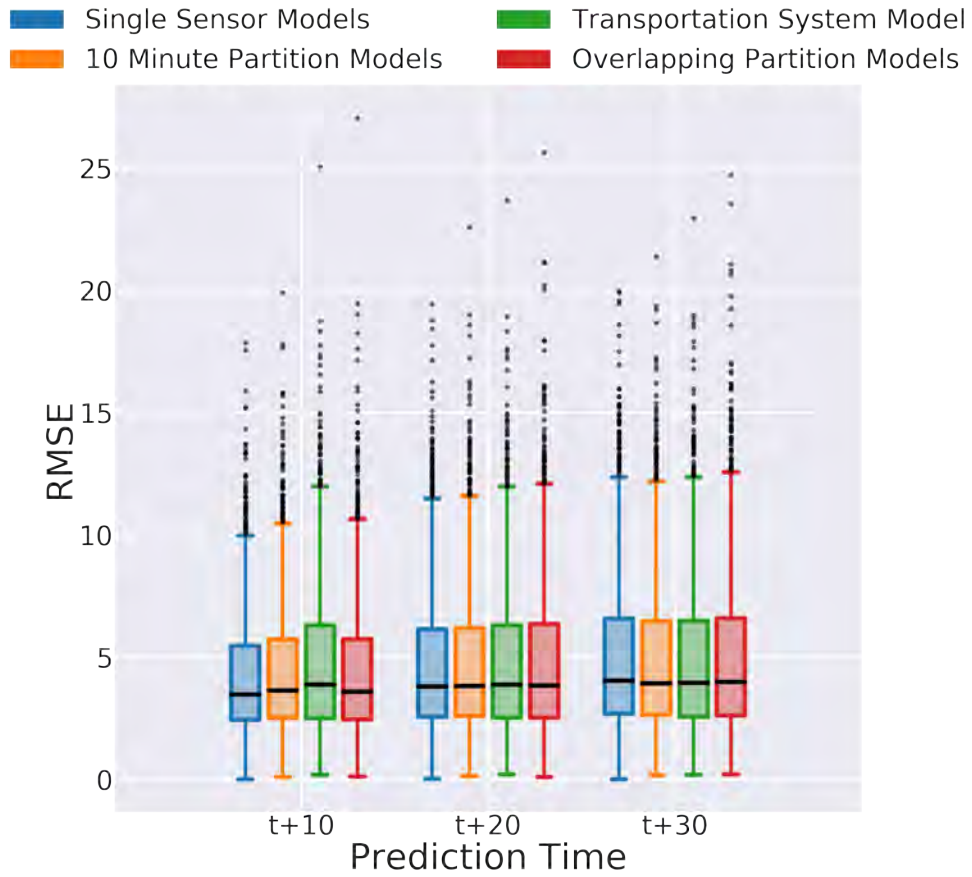


Figure 6.51: Comparison of average RMSE of traffic sensors in the center group in the 10 Minute Partition System Models

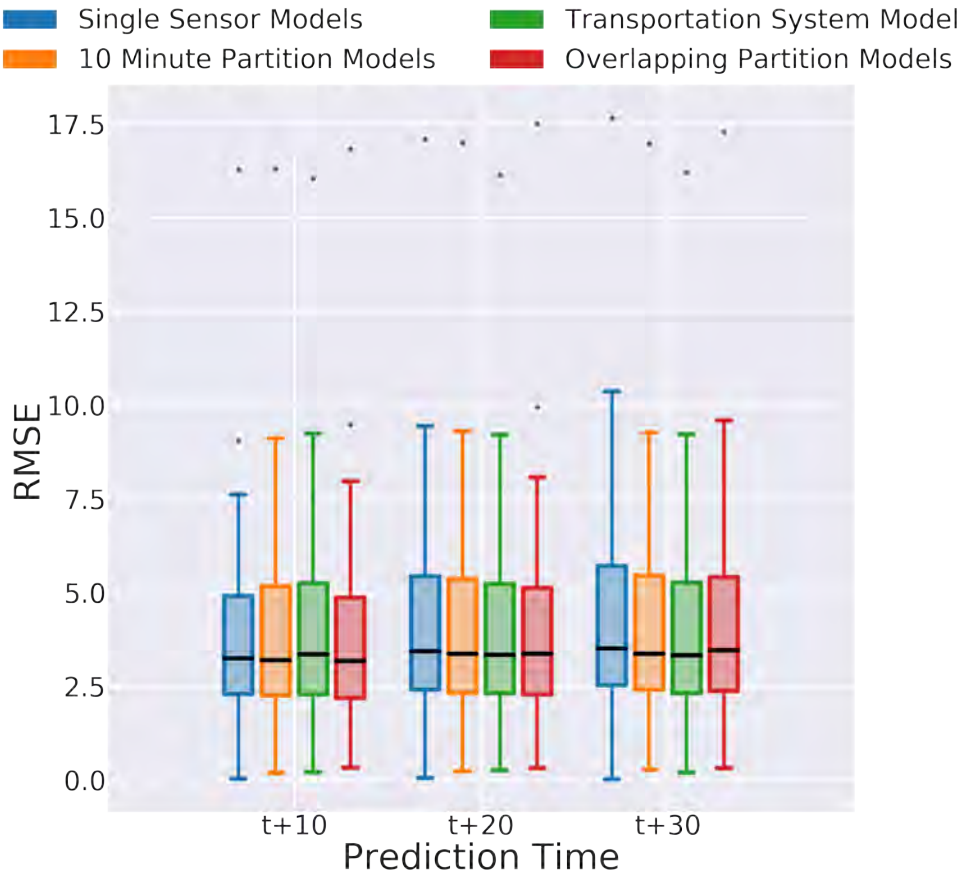


Figure 6.52: Comparison of average RMSE of traffic sensors in the end group in the 10 Minute Partition System Models

5 Minute Partition Group Comparison

The figures 6.53 - 6.55 indicate that the 5 minute partition models give the best average prediction accuracy compared to the other three models for all of traffic sensor groups, i.e. start, center and end group.

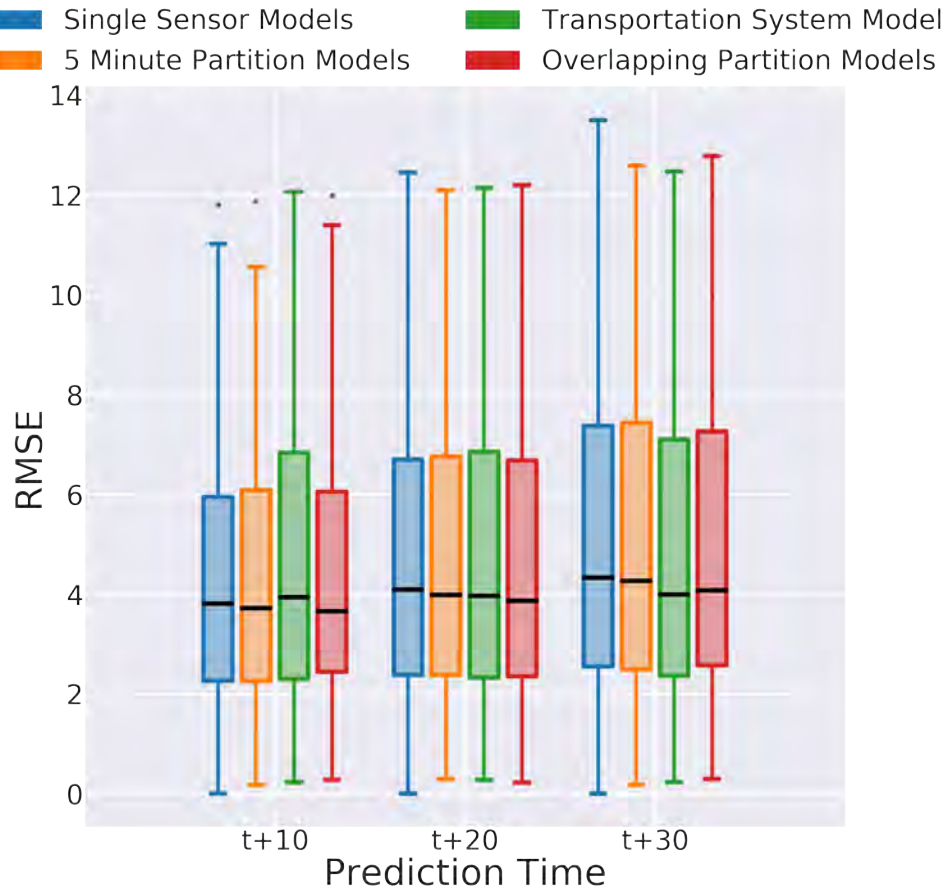


Figure 6.53: Comparison of average RMSE of traffic sensors in the start group in the 5 Minute Partition System Models

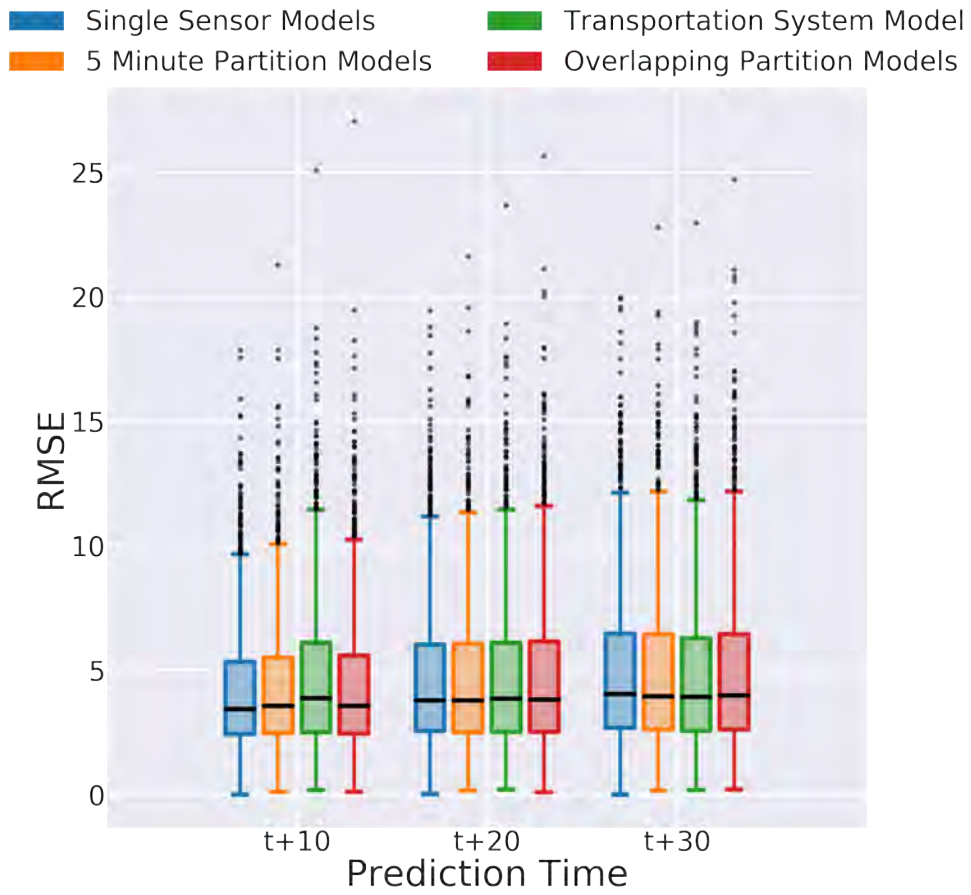


Figure 6.54: Comparison of average RMSE of traffic sensors in the center group in the 5 Minute Partition System Models

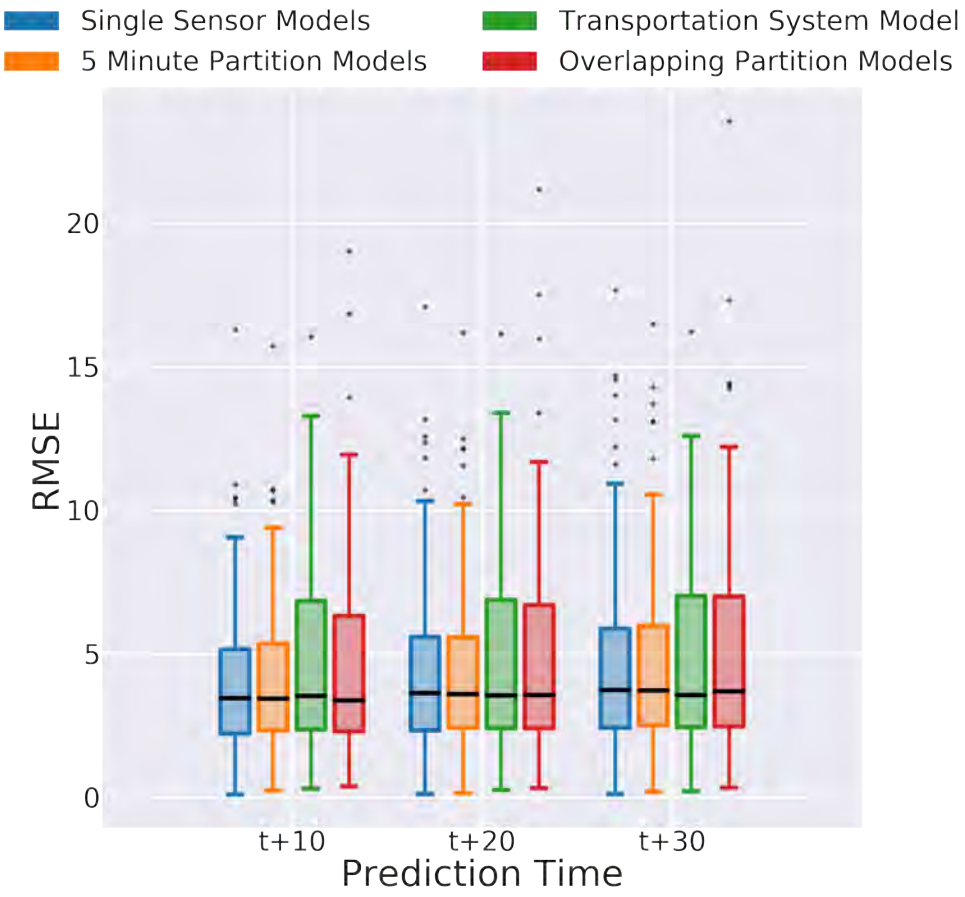


Figure 6.55: Comparison of average RMSE of traffic sensors in the end group in the 5 Minute Partition System Models

3 Minute Partition Group Comparison

The figures 6.56 - 6.58 indicate that the 3 minute partition models give the best average prediction accuracy compared to the other three models for all of traffic sensor groups, except for the start group where the single sensor model has the lowest root mean squared error.

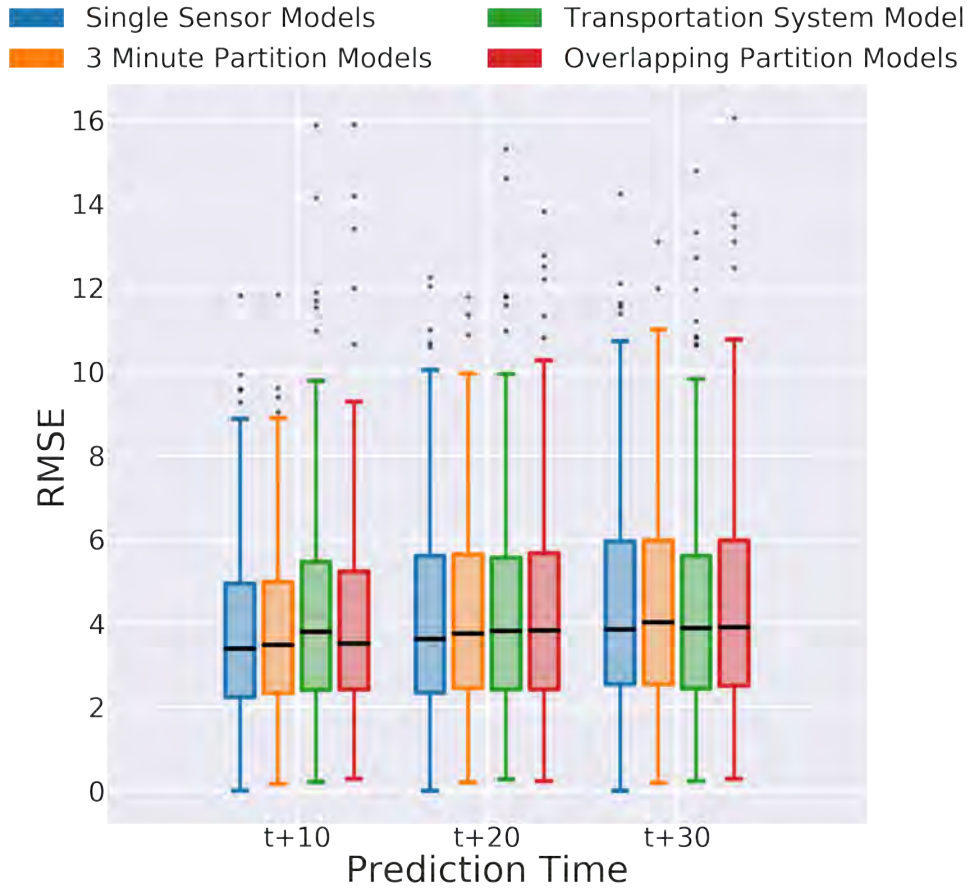


Figure 6.56: Comparison of average RMSE of traffic sensors in the start group in the 3 Minute Partition System Models

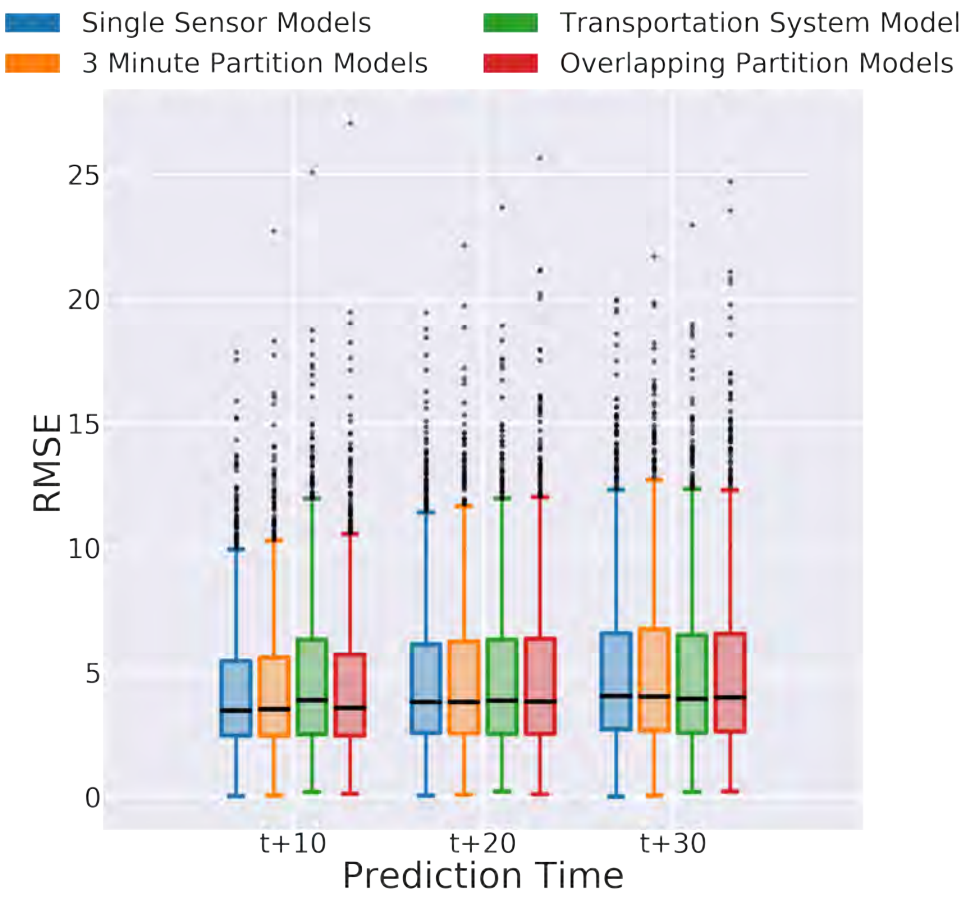


Figure 6.57: Comparison of average RMSE of traffic sensors in the center group in the 3 Minute Partition System Models

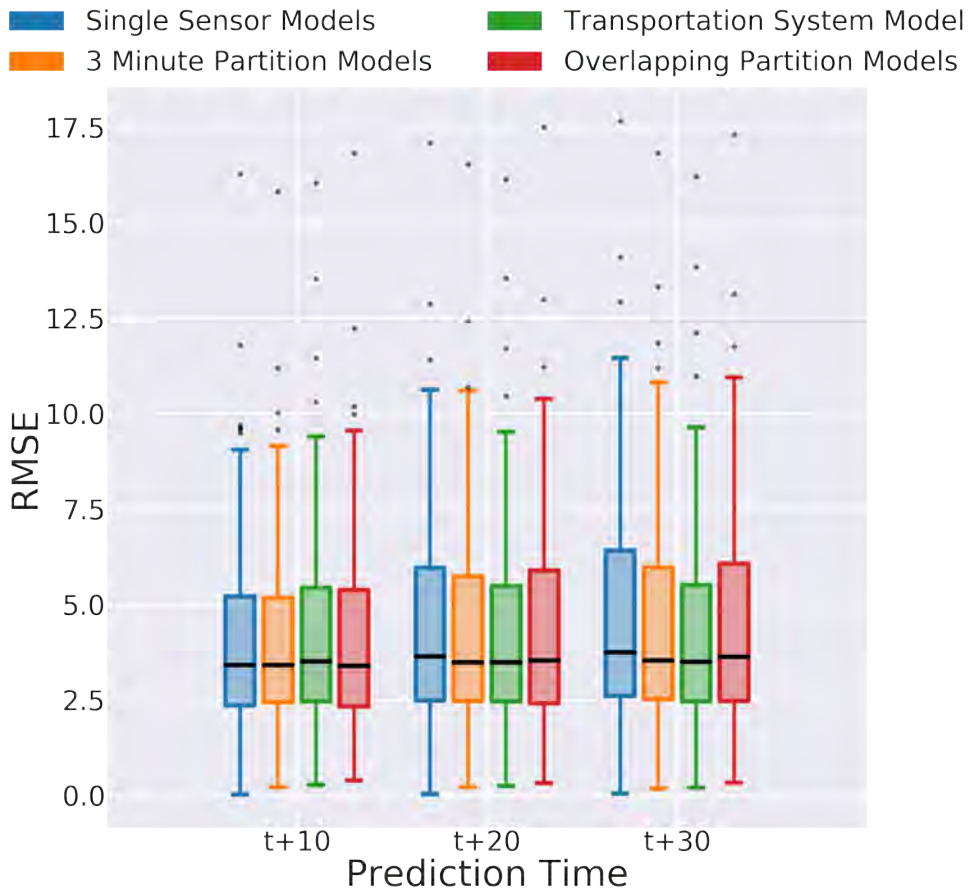


Figure 6.58: Comparison of average RMSE of traffic sensors in the end group in the 3 Minute Partition System Models

Chapter 7

Conclusion and Future Work

The purpose of the thesis was to design and implement a graph consisting of traffic sensors located in Stockholm and Gothenburg, a sequential weight based graph partitioning algorithm, and four types of traffic density prediction models.

Past traffic prediction research projects have not explored how to address scalability and performance issues related to growing transportation infrastructure without compromising the number of traffic sensors. Examples of these issues are related to traffic sensor datasets becoming increasingly larger and more complex as the transportation system grows. This forces data scientists to create larger and more complex neural networks to model the non-linear and stochastic characteristics of traffic flow. These complications may affect accuracy, training, and prediction time. Another issue that arises from an expanding transportation infrastructure is the growing volume of data sent over a computer network to a central data processing center that requires expansion of the computer network to deal with escalating bandwidth and latency problems.

Table 6.7 shows that it is possible to reduce the complexity of the model, the amount of input data to the model, the time used to train the model, and the time it takes to produce model predictions. At the same time, the standard deviation of the root mean squared error between the models ranges only between 0.06 to 0.2 for each of the future predictions.

In this thesis, a solution to address these scalability and performance issues has been developed, implemented and tested by means of experimentation. This solution enables division of the transporta-

tion system into smaller and more manageable partitions based on travel time distance for traffic predictions. These partitions can then be distributed closer to the physical location of the traffic sensors or in other words to the edge of the transportation system instead of using a central data processing system. Traffic predictions produced from these smaller partitions located close to the sensors can then be sent to commuters, who are traveling within the location of the partition. This reduces both the time of producing predictions and sending them since the models are smaller and require less data for each prediction. Local data processing systems responsible for a partition can compress and send predictions to a central data processing system to accumulate a system-wide prediction for the whole transportation system.

The contribution of the thesis are in line with the problem definition in section 1.1 and the four research questions stated in section 1.2. The thesis gave a detailed description of the design and implementation of a weighted directional graph based on the traffic sensor information provided by Trafikverket. The resulting graph structure was evaluated by means of plotting each traffic sensors' site vertex and each edge representing a road connecting two sites on a geographical map. The graph structure represented adequately the physical world as all of the vertices matched with its corresponding traffic sensors' site and all roads connecting sites were represented by an edge.

Three types of sequential weight based graph partitioning algorithms were designed and implemented as a part of this thesis. The evaluation of the algorithms showed that it is possible to partition a transportation system represented with a graph based on a vehicle travel time. This was done by creating simple evaluation scenarios, where a graph representing a small transportation system was partitioned using one of the algorithms and the results verified to be identical to manually partitioning the graph. Lastly, the constructed graph was partitioned using the proposed algorithms to produce partitions that can be used for the construction of traffic prediction models.

Three types of models for traffic predictions were designed, implemented and compared, i.e. a transportation system model partitioned transportation system models, and single sensor models. These models were trained using data from all of the traffic sensors in Trafikverket's transportation system to produce traffic density prediction models that predict up to 30 minutes into the future. The evaluation of the produced traffic density prediction models showed that traffic den-

sity predictions produced from the partitioned transportation system models are comparable in terms of accuracy to the single sensors models and to the whole transportation system model. Accuracy was determined by calculating the average root mean squared error between predicted values and actual density values for all of the traffic sensors in the transportation system. The root means squared error was in the range 4 to 5 indicating that overall prediction accuracy between the models are similar.

This leads to the conclusion that partitioning a transportation system into smaller units is a viable solution to problems of scalability in a growing transportation system.

A forth type of traffic prediction model was designed and implemented to use overlapping partitions of a transportation system to increase the average traffic prediction accuracy. Overlapping partitions involve predicting for a group of traffic sensors using spatial and temporal traffic information from surrounding traffic sensors. Feeding the models with enough information about the surrounding partitions should produce predictions with higher accuracy. This was supported by the results of the experiments in the thesis. The average root mean squared error for this type of model is lower for traffic predictions over 10 minutes as compared to the first three models.

To conclude, the results of the thesis demonstrate that partitioning a transportation system is a viable solution to produce traffic prediction models as the average prediction accuracy for each traffic sensor across the different types of models are comparable. This solution tackles scalability issues produced by increased deployment of traffic sensors in a transportation system by reducing the model complexity and the number of traffic sensors each prediction model is responsible for. A more decentralized and effective solution can be achieved since the models can be distributed to the edge of the transportation system, i.e. near the physical location of the traffic sensors, reducing prediction and response time of the models.

7.1 Limitations

Even though research is carefully planned and prepared, limitations and shortcomings can always be found. Researchers who intend to make use of the results of the work in this thesis need to be aware of

the following.

As with many projects, the quality of the results depends on the quality of the dataset and its usage in various steps of the research process. The three parts of the thesis involving the construction of a weighted traffic sensor graph, sequential weight based partitioning algorithms, and traffic density prediction models are built on top of each other and thereby interdependent. Hence, the results of each research part depends on the preceding part that in turn will affect the final results of thesis.

As for the quality of the weighted graph produced in the first part of the thesis, it depends on the quality of the sensor information dataset provided by Trafikverket and the edge weight calculations for travel time between sensors.

In addition, the sequential weight based partitioning algorithm depends directly on the correctness of the provided weighted graph and how thoroughly the algorithm was tested and verified.

Finally, the traffic density prediction models depend on the quality of the partitions produced by the partitioning algorithm, the correctness of the algorithm, and the accuracy of the graph that is being partitioned.

7.2 Future Work

The work in this thesis can be extended in several ways as discussed in the following sections.

7.2.1 Sensor Graph Refinement

Firstly, the weighted graph was constructed with sensor information from Trafikverket and edge weight calculations using the average vehicle speed during rush hour and the recorded length between sensors or GPS coordinates. This may be improved by using other hours of the day to find the average vehicle speed that is more representative of the traffic flow and travel time between sensors or by measuring the travel time between sensors while commuting through the transportation system. Another possible improvement is to verify and correct sensor locations, where the GPS coordinates are positioned off-roads. More correct GPS coordinates for sensor locations will improve the accuracy of travel time distance for edges in the sensor graph.

7.2.2 Graph Partitioning Improvements

It is possible to improve the sequential weight based graph partitioning algorithm by parallelizing the algorithm to partition parts of the graph concurrently and allowing it to run on a distributed computation cluster. This improvement could be done by detecting disconnected components of the graph and executing the partitioning algorithm on each component in parallel. Changes of this kind can enhance the speed and scalability of the algorithm such that it can handle larger sensor graphs. Yet another improvement would be to evaluate the correctness of the current algorithm using more extensive tests and edge cases. The findings can then be used to refine the algorithm.

7.2.3 Deep Learning Exploration

The traffic density prediction models can be developed further to improve the overall accuracy of predictions and lower the prediction error. This can be done through a more extensive hyperparameter tuning procedure. Experiments using GRU hidden units instead of LSTM have resulted in increased prediction accuracy as was shown in [6]. In addition, a deeper neural network that has more hidden layers may be able to better generalize the nature of traffic flow and increase the prediction accuracy. Exploration of the usage of fully connected layers may improve the models capability of learning the spacial and temporal patterns in traffic flow data as [5] suggests. Investigation of stacked bidirectional and unidirectional long short-term memory layer may improve further on the prediction accurate, as was shown in [33]. It should be explored how a longer prediction horizon effects the accuracy of the models, e.g. 40, 50, 60 minutes into the future.

7.2.4 Model Training

The experiments in the thesis were affected by computational load on the *HopsWorks* cluster originating from various non-controllable sources. Consequently, measurements of the training and prediction time of each model varied considerably. It is, therefore, advisable to run the experiments in an clean environment or without influences from outside factors in order to produce more accurate measurements of the training and prediction time.

7.2.5 Parallel Training

During the thesis work, sequential training of the models was carried out. To achieve faster training, one could use the tools provided by *HopsWorks* to execute the training on multiple GPUs in parallel.

7.2.6 Distributed Traffic Prediction System

The proposed distributed traffic prediction system in this research can by implemented into practice to evaluate the benefits in communication costs and prediction speed as compared to using a centralized traffic prediction system.

7.2.7 Federated Learning

A traffic prediction system constructed from a partitioned transportation system can be augmented by using a federated learning approach proposed in Google's AI blog. Instead of using different traffic prediction models for each partition, a federated learning approach could be applied where the partitions of the transportation system share a prediction model that is used collaboratively to train and improve the model.

Bibliography

[1] Shaojun Zhang et al. "Fine-grained vehicle emission management using intelligent transportation system data". eng. In: *Environmental Pollution* 241 (2018), pp. 1027–1037. ISSN: 0269-7491.

[2] *Transportation Statistics Annual Report 2017*. U.S. Department of Transportation, Bureau of Transportation Statistics, 2017, pp. v, vi, 1–1, 1–4, 1–8, 2–1, 4–1, 5–1, 6–1, 7–1.

[3] Agachai Sumalee and Hung Wai Ho. "Smarter and more connected: Future intelligent transportation system". eng. In: *IATSS Research* 42.2 (2018), pp. 67–71. ISSN: 0386-1112.

[4] Xiaolei Ma et al. "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data". eng. In: *Transportation Research Part C* 54.C (2015), pp. 187–197. ISSN: 0968-090X.

[5] Xingyuan Dai et al. "DeepTrend: A Deep Hierarchical Neural Network for Traffic Flow Prediction". In: (2017).

[6] Rui Fu, Zuo Zhang, and Li Li. "Using LSTM and GRU neural network methods for traffic flow prediction". eng. In: IEEE, 2016, pp. 324–328. ISBN: 9781509044238.

[7] Anne Håkansson. "Portal of Research Methods and Methodologies for Research Projects and Degree Projects". In: 2013, pp. 67–73. ISBN: 1-60132-243-7.

[8] European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. Apr. 2016. URL: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32016R0679 (visited on 07/05/2018).

[9] *Annual Report 2017*. Trafikverket - The Swedish Transport Administration, 2017, pp. 21, 22, 23.

[10] United Nations. *Sustainable Development Goals | UNDP*. Jan. 2016. URL: http://www.undp.org/content/undp/en/home/sustainable-development-goals (visited on 07/05/2018).

[11] Luz Elena Y Mimbela et al. "Summary of Vehicle Detection and Surveillance Technologies used in Intelligent Transportation Systems". In: *Federal Highway Administration's Intelligent Transportation Systems Program Office* (Nov. 2000), pp. 3-1 –3-2.

[12] Luz Elena Y Mimbela et al. "Summary of Vehicle Detection and Surveillance Technologies used in Intelligent Transportation Systems". In: *Federal Highway Administration's Intelligent Transportation Systems Program Office* (Nov. 2000), pp. 4-4 –4-6.

[13] Lawrence A Klein et al. *Traffic Detector Handbook. Third Edition — Volume I*. Tech. rep. FHWA-HRT-06-108. United States. Federal Highway Administration, 2006, pp. 2–1.

[14] Lily Elefteriadou. *Springer Optimization and Its Applications. An Introduction to Traffic Flow Theory*. Vol. 84. Springer-Verlag New York, 2014, pp. 68–69. ISBN: 9781461484349.

[15] Luz Elena Y Mimbela et al. "Summary of Vehicle Detection and Surveillance Technologies used in Intelligent Transportation Systems". In: *Federal Highway Administration's Intelligent Transportation Systems Program Office* (Nov. 2000), pp. 5-9 –5-15.

[16] Lawrence A Klein et al. *Traffic Detector Handbook. Third Edition — Volume I*. Tech. rep. FHWA-HRT-06-108. United States. Federal Highway Administration, 2006, pp. 1-15 – 1-17 & 2-79 –2-83.

[17] Lawrence A Klein et al. *Traffic Detector Handbook. Third Edition — Volume I*. Tech. rep. FHWA-HRT-06-108. United States. Federal Highway Administration, 2006, pp. 1-14 – 1-15 & 2-68 –2-70.

[18] Luz Elena Y Mimbela et al. "Summary of Vehicle Detection and Surveillance Technologies used in Intelligent Transportation Systems". In: *Federal Highway Administration's Intelligent Transportation Systems Program Office* (Nov. 2000), pp. 5-1 –5-8.

[19] Lily Elefteriadou. *Springer Optimization and Its Applications. An Introduction to Traffic Flow Theory*. Vol. 84. Springer-Verlag New York, 2014, pp. 61–65. ISBN: 9781461484349.

[20] Lily Elefteriadou. *Springer Optimization and Its Applications. An Introduction to Traffic Flow Theory*. Vol. 84. Springer-Verlag New York, 2014, pp. 69–72. ISBN: 9781461484349.

[21] Lily Elefteriadou. *Springer Optimization and Its Applications. An Introduction to Traffic Flow Theory*. Vol. 84. Springer-Verlag New York, 2014, pp. 76–78. ISBN: 9781461484349.

[22] Robin J Wilson. *Introduction to graph theory. Fourth Edition*. Addison Wesley Longman Limited, 1996, pp. 1–6, 8–14, 26–30. ISBN: 0582249937.

[23] Narsingh Deo. *Graph theory with applications to engineering and computer science. First Edition*. Dover Publications, 2016, pp. 1–2, 7, 11–13, 229–233. ISBN: 9780486807935.

[24] Richard J Trudeau. *Introduction to graph theory. Second Edition*. Dover Publications, INC, 1994, pp. 18, 19, 24, 41, 105. ISBN: 97804 86678702.

[25] M.E.J. Newman. *Networks: An Introduction. First Edition*. Oxford University Press Inc., 2010. Chap. 11, pp. 337–341. ISBN: 97801992 06650.

[26] Andreas C Müller, Sarah Guido, et al. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[27] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. " O'Reilly Media, Inc.", 2017.

[28] Christopher Olah. *Understanding LSTM Networks*. Aug. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs (visited on 07/05/2018).

[29] C.P.IJ. Van Hinsbergen, J.W.C. Van Lint, and F.M. Sanders. "Short Term Traffic Prediction Models". In: *14th World Congress on Intelligent Transport Systems: ITS for a Better Life, Beijing: Research Institute of Highway, Chinese Ministry of Communications*. Department of Transport & Planning, Delft University of Technology. Oct. 2007.

[30] J.W.C. Van Lint and C.P.IJ. Van Hinsbergen. "Short-term traffic and travel time prediction models". In: *Artificial Intelligence Applications to Critical Transportation Issues* 22.1 (2012), pp. 22–41.

[31] J.W.C. Van Lint, Henk J. Van Zuylen, and H. Tu. "Travel time unreliability on freeways: Why measures based on variance tell only half the story". In: *Transportation Research Part A: Policy and Practice* 42.1 (2008), pp. 258–277.

[32] Nicholas G. Polson and Vadim O. Sokolov. "Deep learning for short-term traffic flow prediction". eng. In: *Transportation Research Part C* 79.C (2017), pp. 1–17. ISSN: 0968-090X.

[33] Zhiyong Cui, Ruimin Ke, and Yinhai Wang. "Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction". In: (2018).

[34] Zainab Abbas et al. "Short-Term Traffic Prediction Using Long Short-Term Memory Neural Networks". In: *2018 IEEE International Congress on Big Data (BigData Congress)*. IEEE Computer Society Digital Library. 2018, pp. 57–65.

Appendix A

Source Code and Software

The source code for the thesis can be found using the following URL:
https://github.com/reginbald/kth-master-thesis-project

The following software was used in this thesis:
Python 2.7 an interpreted high-level programming language.
https://www.tensorflow.org/
NetworkX a Python library for studying graphs and networks.
https://networkx.github.io/
PyCharm an integrated development environment.
https://www.jetbrains.com/pycharm/
Jupyter a web-based interactive computational environment for the creation of Jupyter notebooks.
https://jupyter.org/
Apache Spark a cluster-computing framework for large-scale data processing.
https://spark.apache.org/
TensorFlow a library for high performance numerical computation with strong support for machine learning and deep learning.
https://www.tensorflow.org/
Keras a high-level neural network library that is able to run on top of Tensorflow.
https://keras.io/
HopsWorks a platform that integrates some of the most popular services for Data Science and Data Engineering.
https://www.hops.io/

TRITA TRITA-EECS-EX-2018:765